INTRODUCTION

Part II of this Engineering Product Specification (EPS) is essentially a detailed description of the GE-645 Prototype Processor hardware, both physically and electrically. After Section 7.0 which describes the physical configuration of the Processor, the theory of operation for the Processor and the individual functional units that make it up are explained, then the detailed operation.

EPS-I for the GE-645 Prototype Processor must be familiar to the reader before attempting to read EPS-II, since the material presented in the following pages is but a logical explanation of the philosophy stated in EPS-I.

7.0 PHYSICAL CONFIGURATION

The GE-645 Prototype Processor is contained in a single double size cabinet (freestanding or one wing of a X or Y configuration). One end of the cabinet (G.E. Dwg. 43E160145) mounts the Maintenance Panel which is exposed to the operator. At the rear of the Maintenance Panel are the processor fault vector switches and memory interlace controls. A complete description of the Maintenance Panel is given in Part I Section 3.0 of EPS-I. The opposite end of the Processor cabinet provides jack connectors (snowplow) for up to eight possible memory controllers, an elapsed time meter, power supplies for relay power, the primary input power circuit breaker, and a relay panel. This panel mounts the relays which select combinations of the approximately 1500 sampling points that are displayed on the Maintenance Panel indicators. All power supplies for the six frames of circuit cards are located in the center of the cabinet and behind the Maintenance Panel. The supplies for the Maintenance Panel display lamps are also mounted on brackets behind the Maintenance Panel. Voltage regulation dropping resistors for the various power supplies are mounted on the top of the cabinet.

The general layout of the Processor cabinet is given in paragraph 1.5 of EPS-I. For a more detailed explanation of the physical layout, refer to the cabinet mechanical drawings referenced from the top assembly drawing and P.L., 43E160145.

8.0 OPERATIONAL DESCRIPTION

(To be supplied at a later date and will contain a top level discussion of total Processor operation: Operations Unit operations (loads, stores, etc.) Control Unit operations (special instructions, transfers, etc.) and how the Base and Interface Frames and associative memory are utilized to fetch instructions and operands. For the present, the only top level discussions are in EPS-I, Sections 1 and 2 and are not adequate for a general picture of the entire Processor.)

9.0 CONTROL UNIT DETAILED THEORY

This section covers the detailed theory of operation of the Control Unit. The Control Unit consists of one frame of logic boards in the Processor cabinet designated the Control Frame. For the purpose of this discussion the names Control Unit and Control Frame are used interchangeably. The information in this section covers a brief general description of the Control Frame; the modes of operation generalized into classes of instruction that cause the same operating sequence; the principle hardware elements and functions; and the basic operating sequences that take place to perform the functions.

9.1 GENERAL DESCRIPTION

The major functions of the Control Frame are:

To perform all Processor control functions

To perform address modification when specified by the instruction or indirect word.

To define Processor mode of operation (Master/Slave, Absolute/Append)

To perform external interrupt recognition and servicing (including faults)

To perform operation decoding

To buffer a total of four instructions in the Processor at one time

Operation code buffering for two additional instructions that have the operand addresses prepared and are waiting to be accepted by the Operations Unit

At the present time, the meaning of the aforementioned major functions may appear vague, but the following paragraphs will elaborate all the functions in greater detail.

9.1 - contd

To perform the basic functions listed above requires almost a full complement of component boards in the Control Frame. Since most of the control frame logic deals with control functions, the block diagram (43D160212) and the control diagrams (43D163387) cannot show neatly defined packages of logic that perform well defined functions. Therefore, the discussion in the following paragraphs is presented in this manner:

1) Modes and sequences of operation within the Control Frame are grouped according to the similarity of the sequences involved in executing the specific instruction beginning in paragraph 9.2. Modes and sequences of operation that cannot be generalized, are explained in detail.

2) The principle hardware elements and functions that can be defined physically or functionally are explained in detail beginning in paragraph 9.3.

3) The interrelationship of the various functions and hardware elements, and how they perform the basic functions in each mode and sequence of operation, is covered in detail beginning in paragraph 9.4.

Presumably the reader will be able to supply the finer details of the logical operation by reference to the applicable logic prints and diagrams referenced during the following discussions.

9.1.1 INTERFACE LINE SUMMARY

All interface signals between the Control Frame and other frames within the Processor are listed by EDA mnemonic in Tables 9.1.1 and 9.1.1a. The total interface between all frames in the Processor is listed in drawing 43A164138.

TABLE 9.1.1 Control Frame Inputs

Mnemonic		Description
		Arithmetic Frame
DFNRC	($\overline{FNR}$)	FLAG TO CF-OU READY FOR DIRECT OPERAND
DFNYC	($\overline{FNY}$)	FLAG TO CF-M REG. RDY FOR OPERAND
DRGSC		GS BUT NOT GF
DRIJC		A TERMINATE REPEAT MODE CONDITION HAS OCCURRED
DTASC		TEST AND STEP
EN18C-EN35C		SCANNER LOGIC
FE00C-FE07C	(E0-7)	8-BIT EXPONENT REGISTER
FI00C-FI09C	(IND0-9)	INDICATOR REGISTER
QSLTC	($LAST)	
QZGGC		GENERAL STROBE
QZZAC	($DVCK)	DIVIDE CHECK FAULT PULSE
QZZOC	($FOLF)	OVERFLOW FAULT PULSE
		Register Frame
DDACD		XFER CNTRL. FOR E REG. 0-7 TO ZDO 0-7
F000D-F017D	(X0 0-17)	18-BIT INDEX REGISTER NO. 0
F100D-F117D	(X1 0-17)	18-BIT INDEX REGISTER NO. 1
F200D-F217D	(X2 0-17)	18-BIT INDEX REGISTER NO. 2
F300D-F317D	(X3 0-17)	18-BIT INDEX REGISTER NO. 3
F400D-F417D	(X4 0-17)	18-BIT INDEX REGISTER NO. 4
F500D-F517D	(X5 0-17)	18-BIT INDEX REGISTER NO. 5
F600D-F617D	(X6 0-17)	18-BIT INDEX REGISTER NO. 6
F700D-F717D	(X7 0-17)	18-BIT INDEX REGISTER NO. 7
FA00D-FA71D	(AQ 0-71)	72-BIT AQ REGISTER (0-35 A)(36-71 Q)
FDPFD		INDICATES LWR. 36 BITS DBL. P. OPRND. NEXT ON ZDI

TABLE 9.1.1 Control Frame Inputs (contd)

Mnemonic	Description
	Register Frame
FM28D	MEMORY OPERAND BIT 28-APPENDING MODE
QZOBD ($QR)	O-REGISTER STROBE
QZSSD ($START)	START STROBE
QZSZD ($SDY)	OU STORE DATA READY TO STORE
QZYXD ($YRY)	OU READ FOR NEW OP CODE
	Base Frame
BCLTF ($\overline{\text{FLT}}$)	RECOGNIZE FAULT TAG 1-3
BEOTF ($\overline{\text{BIT 29+ITS+ITB}}$)	
BFRQF ($\overline{\text{FRQ}}$)	GROUP 6 FAULT PRESENT
BLPCF	LOW PRIOR. FLTS/$ABT
BONCF (GP2 PONC)	GP2 AND PONC
BPETF (PBR=TBR)	CONTENTS OF PBR EQUALS TBR CONTENTS
BRROF ($\overline{\text{RR}}$)	RELEASE $R
BS03F (RS3)	RESET GROUP 3,4,5,6 FAULTS
BSBTF (SABT)	START ABORT SEQUENCE
BSFFF ($\overline{\text{SFF}}$)	STOP FOR FAULTS
DFA0F-DFA4F (FAA-E)	FAULT ADDRESS ENCODES (FA0 MOST SIG.)
DI00F-DI35F (ZDI0-35)	DATA INPUT BUS
DOMRF	MEMORY READY
DZ00F-DZ17F (ZBRI0-17)	BASE REGISTER INTERNAL BUS
EDPIF (DPI)	DOUBLE PI CYCLE
EFOCF (FOC)	FAULTY OP. CODE FAULT
EGBOF (GBO)	GENERAL BASE INSTRUCTION F/F
EINAF	INITIALIZE
EN03F-EN05F (INC3-5)	INCREMENTOR REGISTER

TABLE 9.1.1 Control Frame Inputs (contd)

Mnemonic	Description
	Base Frame
EPEOF ($\overline{\text{PEO}}$)	PARITY ERROR IN OPERAND
EPFIF ($\overline{\text{PFPI}}$)	PARITY ERROR IN BUFFRD. INSTRCN.
ERSTF	REMOTE STOP F/F
ESEPF (SEP)	PRTY, ERR. IN OU OPERAND
ESNOF (SNO)	SNAPSHOT ZERO F/F
ESN1F (SN1)	SNAPSHOT ONE F/F
EWCCF ($\overline{\text{WC}}$)	WAIT FOR EXECUTE INTRPT. CELL NO. CYCLE
EWFOF (WF)	ABORT F/F
EWFBF (WFTB)	WAIT FOR TROUBLE FLIP-FLOP
QABTF ($ABT)	END OF WF PULSE
QSBTF	START ABORT PULSE
QSNOF ($SNO)	END OF SNAPSHOT 0 PULSE
QSNZF ($SN1)	END OF SNAPSHOT 1 PULSE
QZBCF ($BC)	DATA IS READY
	Interface Frame
BDPCM	SET DPIN F/F
BFAAM ($\overline{\text{FA}}$)	FINAL ADDRESS FETCH (FAPM+FANP+ABS)
BIAOM-BIA3M (IBAO-3)	INTERRUPT BLOCK ADDRESS BITS 0-3
BLSRM	INT BLOCK ADDRESS BIT
BSCAM	DELAYED TRIGGER
BSCHM	SCOPE GATE
DAIPM-DHIPM ($\overline{\text{IP0-7}}$)	EXTERNAL INTERRUPT PRESENT-PORTS A-H
EMS2M (MSF2)	NEXT INST. PR. MASTER/SLAVE MODE BUFFER
EMLSM	MULT. LOAD/STORE INSIGN. FLAG
ERDYM (SRDY)	PROC. READY FOR STEP FLAG

TABLE 9.1.1 Control Frame Inputs (contd)

Mnemonic	Description
	Interface Frame
FN00M-FN17M (SCA0-17)	SCANNER BITS 0-17
MA15M-MA17M (RS15-17)	RS-ADDER BITS 15-17
QDSTM ($DS)	DATA STORED PULSE
QIAZM ($IAZ)	
QIFFM ($IFF)	INTERFACE FRAME FINISHED PULSE
QINTM ($INT)	GENERAL $INT
QOPNM ($PIN)	
QPRIM ($PRI)	
QSDAM ($DA)	
QSRFM ($RSRF)	APPNDG. FETCH STARTED PULSE
QSSFM ($SSRF)	FINAL FETCH STARTED PULSE
	Maintenance Panel
SASMT	ASSOC. MEMORY OFF SWITCH
SCOOT	OU/CU OLAP OFF SWITCH
SI00T-SI35T	36-BIT INSTRUCTION WORD SWITCHES
SQC1T	CU TEST CLOCK SWITCH
SREXT	REPEAT EXECUTE SWITCH
SSMT	STEP MEMORY SWITCH
SSOFT	OFF SWITCH
SSOPT	STOP SWITCH
SSPGT	STEP PROGRAM SWITCH
STMRT	REAL-TIME/MEM. CYCLE SWITCH FOR TIMER REGISTER
SSTPT	STEP SWITCH
STSTT (TEST)	TEST SWITCH
SXECT	EXECUTE BUTTON
SXSWT	EXECUTE SWITCHES

TABLE 9.1.1a Control Frame Outputs

Mnemonic	Description
	Arithmetic Frame
BRDRB ($\overline{RDR}$)	REPEAT DOUBLE RETURN
BRLRB ($\overline{INT}$)	REPEAT OR LINK RETURN
BRSTB ($\overline{RST}$)	RETURN SENSE TERMINATION
DFCTB (TALLY)	
DTHAB (FL) (PA)	PREP. ADDRESS BUT NOT REPEAT LINK
DTHCB ($\overline{CRESET\ OVFL}$)	RESET OVERFLOW
DTHDB ($\overline{CRESET\ EOFL}$)	RESET EXPONENT OVERFLOW
DTHEB ($\overline{CRESET\ EUFL}$)	RESET EXPONENT UNDERFLOW
DYSZB (YS=0)	Y-ADDER OUTPUT IS ZERO
DZ06B-DZ17B	INSTRUCTION DECODE TO ADDRESS ADDER
EI10B	MASTER MODE INDICATOR
EPAFB	PREPARE FIRST ADDRESS CYCLE
EPIFB	PREPARE INST. ADDRESS
EPNFB	REPEAT MODE INDIRECT CONTROL
EPRFB	PREPARE REWRITE CYCLE FLAG
EPTFB	PREPARE IT ADDRESS
EPZFB	PREPARE SUBSEQUENT ADDRESS CYCLE AFTER PA INDIRECT
QZRAB ($RESET ABORT)	RESET ABORT STROBE
QZSAB ($SET ABORT)	SET ABORT STROBE
	Register Frame
DC00B-DC08B ($\overline{ZORO-8}$)	OPER. TRANSMIT BUS TO OU
DDAMB ($\overline{PR}$)	CONTROL FRAME IN PR CYCLE
DTHUB ($\overline{FD+FT+FL}$)	ANY REPEAT INSTRUCTION (FT+FD+FL)
DW12B (ZI12)	BYTE SIZE BIT
DW15B-DW17B ($\overline{ZI15-17}$)	BYTE POSITION BITS
EI10B ABS	MASTER MODE INDICATOR

TABLE 9.1.1a Control Frame Outputs (contd)

Mnemonic	Description
	Register Frame
EL00B-EL17B (ICTA0-17)	ADDRESS OF INST. IN EXECUTION
MU00B-MU17B (YS0-17)	ADDRESS ADDER
QRDYB ($DRDY)	DATA READY TO RF STROBE
QZOYB ($ORY)	OP CODE IS READY STROBE
QZPOB ($64 KC)	64-KC STROBE
QZPQB ($BYTE)	BYTE POSITION REGISTER STROBE
QZPXB ($X0-7)	INDEX REGISTERS STROBE
	Associative Memory Frame
MB00B-MB11B (BS0-11)	ADDRESS BASE ADDER
QSTAB ($)	START ASSOC. MEMORY PULSE
QDMFB	STROBE DIRECT OPERAND
QESRB	PREPARE INTERRUPT STROBE
QSCAB	STROBE TO SCANNER LOGIC
	Base Frame
BBCRB (BCR)	INSTRUCTION IS REPEAT OR STORE
BCDBB (CDB0-17/)	CONTROL FOR DB-REG./ZDIBF0-17
BCTAB (CT)	TAG REG. MODIFICATION IDC OR DIC
BDDBB (CDB0-17/)	CONTROL FOR DB-REG./ZDIRF18-35
BDPIB (SPI)	DECODE=SPI=DPI AND EXI AND RPI
BEADB (EA)	PREPARED ADDRESS IS FOR OPERAND
BHLTB (HALT)	LOGIC DECODE FOR STOP
BICFB (CSETIC)	LEVEL TO SET IC FLIP-FLOPS
BICRB (CRESET IC)	LEVEL TO RESET IC FLIP-FLOPS
BIPCB ($\overline{PC}$)	INTERRUPT PRIORITY CHANNEL
BIPFB ($\overline{IPF}$)	DECODE LOGIC IPR STROBE
BMPAB (FRA)	FAULT OR EXECUTE INT. IN PA CYCLE
BPNLB (CPNL SW/ZDI)	CONTROL FOR PANEL SWTCHS/ZDIRF0-35

TABLE 9.1.1a Control Frame Outputs (contd)

Mnemonic	Description
	Base Frame
BSPAB (CSET PA)	LEVEL TO SET PA FLIP-FLOP
BSPIB (CSET PI)	LEVEL TO SET PI FLIP-FLOP
BSPNB (CSET PN)	LEVEL TO SET PN FLIP-FLOP
BSPTB (CSET PT)	LEVEL TO SET PT FLIP-FLOP
BSPZB (CSET PZ)	LEVEL TO SET PZ FLIP-FLOP
BSWNB (CSET WN)	LEVEL TO SET WN FLIP-FLOP
BTDLB (TMCH=2)	TOO MUCH COUNTER EQUALS 2
BTDMB (TMCH=1)	TOO MUCH COUNTER EQUALS 1
BTDNB (TMCH=0)	TOO MUCH COUNTER EQUALS 0
BTRGB (TRG01)	TRANSFER GO GROUP SUB11 DECODE
BYCFB (Y)	DIRECT OPERAND THE Y OF IT
DZERB (CZEROES/)	CONTROL FOR ZEROS/ZDIRF0-17
DA00B-DA02B (ZY0-2)	INST. REG. SW. EVEN
DA29B (ZY29)	INST. REG. SW. EVEN
DB00B-DB02B (ZC0-2)	INST. REG. SW. ODD
DB29B (ZC29)	INST. REG. SW. ODD
DW00B-DW08B (ZI0-8)	INST. REGISTER DECODE BUS
DW12B-DW17B (ZI12-17)	INSTRUCTION REGISTER DECODE BUS
ECICB (CIOC)	FF HOLDING INTERFACE TERM WHEN CIOC INST. GIVEN.
EDVKB (DVCK)	DIVIDE CHECK FF
EEXFB (EXF)	EXECUTE FF
EFLFB (FOLF)	FAULT OVERFLOW FF
EKC1B (1KC)	TIMER 1KC OUTPUT
ELSTB (LST1)	
EMSYB (M/S)	MASTER/SLAVE FLAG ICTA8
ERPDB (FD)	REPEAT DOUBLE MODE

TABLE 9.1.1a Control Frame Outputs (contd)

Mnemonic	Description
	Base Frame
ERPLB (FL)	REPEAT LINK MODE
ESPRB	CONTROL FRAME STEP FLAG
ERPTB (FT)	REPEAT MODE
ESTPB (STOP)	STOP FLIP-FLOP
ETERB ($\overline{TERM}$)	TERMINATION OF INSTRUCTION
EWIWB (WI2)	WAIT FOR INST. FOR I-REG.
EXDEB (XDE)	EXECUTE DOUBLE FROM EVEN LOCATION
MB00B-MB17B (BS0-17)	ADDRESS BASE ADDER
QIPRB ($IPR)	STROBE TO SET IPR FLIP-FLOPS
QSCOB ($C)	$C TO BASE FRAME
QZCEB	STROBE EVEN SEG TAG REG
QZCOB	STROBE ODD SEG TAG REG
QZCRB ($CSR)	$CSR TO BASE FRAME
QZSRB ($R)	GENERAL STROBE IN CF
	Interface Frame
BEADB (EA)	PREPARED ADDRESS IS FOR OPERAND
BFSLB ($\overline{RPTS}$)	REPEAT FLAG DECODE FD, FT, OR FL
BIPCB (PC)	INTERRUPT PRESENT
BISIB	INHIBIT START INTERRUPT
BMPAB (FPA)	FAULT OR EXECUTE INT. IN PA CYCLE
BSITB	STOP FOR TERMINATE
BSPAB (CSET PA)	LEVEL TO SET PA
BSPIB (CSET PI)	LEVEL TO SET PI FF
BSPNB (CSET PN)	LEVEL TO SET PN
BSPRB (CSET PR)	LEVEL TO SET PR FF
BSPTB (CSET PT)	LEVEL TO SET PT

TABLE 9.1.1a Control Frame Outputs (contd)

Mnemonic	Description
	Interface Frame
BSWNB (CSETWN)	LEVEL TO SET WN FF
BTBCB	STOP OR FAULT TO SYNC PANEL
BTRGB (TRGO1)	TRANSFER GO GROUP SUB 1 DECODE
BYCFB	DIRECT OPERAND
DD00B-DD35B (ZEOCF0-35)	CONTROL FRAME DATA OUTPUT BUS
DW00B-DW17B (ZI0-17)	INST. REG. DECODE BUS
ECT0B-ECT5B (CT0-5)	TAG BIT STORAGE REGISTER
ECICB	INTERFACE TERM FOR CIOC INST.
EMSYB (MSY)	MASTER/SLAVE FLAG ICTA8
ERP0B-EPR7B (IPR0-7)	INTERRUPT PRIORITY REGISTER
EXSFB (XSF)	EXECUTE SWITCHES FAULT
MB00B-MB17B (BS0-17)	ADDRESS BASE ADDER
QAAMB ($AAM)	START ASSOCIATIVE APPENDING STROBE
QCFFB ($CFF)	COMPARE STROBE
QPRCB ($PRC)	STROBE R GEN CONTROLS
QSABB ($SAB)	START ABSOLUTE STROBE
QSAPB ($SAP)	START APPENDING STROBE
QSCAB ($SCA)	STROBE TO SCANNER LOGIC
SZCCB ($C)	GENERAL STROBE
QZDPB ($DP)	CF DATA AVAILABLE STROBE
	Relay Box
EC00B-EC08B (C00-8)	OPERAND DECODE-ODD
ECT0B-ECT5B (CT0-5)	TAG BIT STORAGE REGISTER
EH00B-EH11B	ICTA PRIME REG
EJ00B-EJ17B (ICTC0-17)	18-BIT Y AND C PAIR ADDRESS INSTRUCTION COUNTER

Cont. on Sh. 9-13

TABLE 9.1.1a Control Frame Outputs (contd)

Mnemonic	Description
	Relay Box
ED00B-EK17B (ICTB0-17)	18-BIT I-BUFFER CONTENTS ADDRESS CONTENTS ADDRESS INST. COUNTER
EL00B-EL17B (ICTA0-17)	18-BIT EXECTN. INSTR. ADDRESS INSTR. COUNTER
EZ10B-EX13B (Z1 0-3)	Z-FUNCTION REGISTER NO. 1
EZ20B-EZ23B (Z2 0-3)	Z-FUNCTION REGISTER NO. 2
FC00B-FC35B (C0-35)	36-BIT INSTRUCTION DECODE REGISTER-ODD
FCE0B-FCE8B (CE0-8)	OPERAND DECODE-EVEN
FI00B-FI71B (ID-71)	72-BIT INSTRUCTION REGISTER
FY00B-FY35B (Y0-35)	36-BIT INSTRUCTION DECODE REGISTER-ODD
MB00B-MB17B (BS0-17)	ADDRESS BASE ADDER
MU00B-MU17B (YS0-17)	18-BIT ADDRESS ADDER
	Maintenance Panel
BHL2B (HALT2)	HALT TO CONTROL PANEL
BIPCB (PC)	INTERRUPT PRIORITY CHANNEL
BJR2B	DECODE FOR TRG02
EBFRB (BFR)	CONTROL FF IN STROBE R GENERATION
ECFRB (CFR)	CONTROL FF FOR STROBE R GENERATION DELAY
EI10B (ABS)	MASTER MODE INDICATOR
EICFB (IC)	XFER DATA TO ZI BUS FF
EIFRB (IFR)	CONTROL FF FOR STROBE R GENERATION
EMF0B (MF0)	MFREE COUNTER
EMF1B (MF1)	MFREE COUNTER
EMSIB (MSI)	MASTER/SLAVE FLAG SUB 2 IN CONTROL FRAME
EMSYB (MSY)	MASTER/SLAVE FLAG ICTA8

TABLE 9.1.1a Control Frame Outputs (contd)

Mnemonic	Description
	Maintenance Panel
EPAFB (PA)	PREPARE FIRST ADDRESS CYCLE
IPIFB (PI)	PREPARE INSTRUCTION ADDRESS
IPINB (PIN)	FF IN BINTB DECODE
EPNFB (PN)	REPEAT MODE INDIRECT CONTROL
EPTFB (PT)	PREPARE IT ADDRESS
EPZFB (PZ)	PREPARE NEXT ADDRESS CYCLE AFTER PA IN INDIRECT
ESDYB (SDY)	FF IN BINTB DECODE
ESREB (SRF3)	SRF3 F/F
ESREB (SRF2)	SRF2 F/F
ETJSB (B)	XFER DATA TO ZOR BUS FF
ETM0B-ETM2B (TMCH0-2)	TOO MUCH COUNTER 0-2
EXDEB (XDE)	EXECUTE DOUBLE FROM EVEN LOCATION
EXDOB (XDO)	EXECUTE DOUBLE FROM ODD LOCATION
	Junction Panel
B999B (MEM. READYC)	MEMORY READY TO CONSOLE
BCPOB	ABSOLUTE MODE TO CONSOLE
BTNLB (START UP)	PROC. READY TO CONSOLE
BWHOB	WAIT FOR INTERRUPT TO CONSOLE
QEARB	PULSE TO CONTROL SPEEDMETER

9.2 MODES AND SEQUENCES OF OPERATION

The modes and sequences of operation within the Control Frame are divided into the following general categories:

1) Mode of Addressing - whether the Processor is in the Absolute or Append mode of addressing for instructions and operands, and what operand address modification is employed.

2) Mode of Execution - whether the Processor is executing instructions in the Master or Slave mode.

In preparing the addresses for instructions and operands, and in the execution of these instructions, the Control Frame goes through a certain sequence of operation. For the purpose of explanation, the sequences have been defined as follows:

1) Instruction fetches - for two or four instructions.

2) Operand fetches - generally dependent on the type of instruction to be executed:

    a) Normal Operations Unit (OU) -- those OU instructions which require an operand to be fetched from, or sent to memory.

    b) Direct Operations Unit -- those OU instructions that do not require a memory access for the operand.

    c) Control Frame (CF) -- those instructions which are executed primarily within the Control Frame.

    d) Base Frame (BF) -- those instructions which involve the Base and/or Associative Memory Frames and generally are executed outside the Control Frame.

    e) Specials -- those instructions which involve multiple frames and individually determine the Control Frame sequence.

9.2 - contd

The sequence of operation is determined by a series of sequence control flags that are set as a function of the decoded instruction, and by the previous and the succeeding sequences.

9.2.1 SEQUENCE CONTROL FLAGS

The timing sequences of the Control Frames are defined by flip-flops called flags. There are four general categories of flags:

1) Prepare
2) Wait for memory
3) Even/odd sequence
4) Auxiliary or special

9.2.1.1 Prepare (P) Flags

The prepare flags are used to define the prepare sequence for instruction, operand, indirect, and indirect then tally word address preparation. These flags are defined as follows:

1) PI - Prepare Instruction (flip-flop). This flag defines with a few exceptions, that period of time that an address for a double-precision instruction fetch is prepared. The PI flag is normally set whenever instructions are to be loaded into the I-register.

2) PA - Prepare Address (flip-flop). This flag defines that period of time during which the preparation of an address for an or the first indirect word specified by the instruction word is accomplished.

3) PZ - Prepare Address of Indirect (flip-flop). This flag defines that period of time during which an operand address is prepared using an indirect word that was pulled from memory as the result of RI or IR modification in the instruction word, (that is, any PA cycle but the first).

4) PN - Prepare Repeat Indirect (flip-flop) defines that period of time used to prepare the operand address from an indirect word "repeated" instructions having

9.2.1.1 - contd

RI modification, and for preparing the transfer address of a return or return double instruction.

5) PT - Prepare IT Operand (flip-flop) defines that period of time during which the preparation of an address using an indirect word brought in as a result of IT modification is accomplished.

6) PR - Prepare Rewrite (flip-flop) defines that period of time during which the address and tally portions of an indirect word brought in as a result of IT modification is updated, and the new updated indirect word is returned to memory for storage.

7) PC - Prepare Execute Interrupt (level) defines that period of time during which an XEC memory command is prepared to acknowledge the presence of an external interrupt. All prepare flags are mutually exclusive except that PC may be set during the first 300 nanoseconds of PA, PZ, PN or PT. The prepare flags are set as a function of strobe $C and the particular prepare operation being performed. Strobe $C in turn is a function of any conditions which are discussed in detail under strobe generation, paragraph 9.3.1.

9.2.1.2 Wait for Memory (W) Flags

The wait for memory flags are used to define the particular wait for memory cycles as follows:

1) W- Wait for Operand (flip-flop) defines that period of time that the Processor is waiting for an operand from memory, or memory is to accept an operand from the Processor.

2) WI_1 - Wait for Instruction 1 (flip-flop) defines that period of time that the Processor is waiting for the memory to send it an even instruction, an odd instruction or an even/odd pair of instructions for the Y- and C-registers.

9.2.1.2 - contd

3) WI_2 - Wait for Instruction 2 (flip-flop) defines that period of time that the Processor is waiting for an even/odd pair of instructions to fill the I-register.

4) WC - Wait for Execute (flip-flop) defines that period of time during which the Processor is waiting for the memory to acknowledge an Execute command which was issued in response to an external interrupt. This flag is located in the Processor fault logic in the Base Frame but is included here since it defines a Control Frame sequence.

5) WF - Wait for Fault (flip-flop) defines that period of time after the presence of a fault has been established, during which the proper conditions are established for entrance into a fault routine. The WF flag is located in the processor fault logic in the Base Frame but is included here since it defines a particular Control Frame sequence.

6) WN - Wait for Indirect (flip-flop) defines the period of time during which the Processor is waiting for an indirect word from Memory.

Flags WI_1, WI_2 and WC are mutually exclusive with each other and with all P flags except that PA may overlap WI_2. The W flag will be on in parallel with a P or a different wait flag, if an Operations Unit operand is being fetched from memory.

9.2.1.3 Even/Odd Sequence Flag

A flag denoted as the IC flip-flop is used to designate whether the Control Frame is working on the even (Y) or odd (C) instruction register. The IC flip-flop is reset for the even instruction and set for the odd instruction. Reset and set conditions are a function of the strobe $C that initiates the beginning of the PA cycle for each instruction.

9.2.1.4 Auxiliary or Special Flags

Within the Control Frame are a number of flags that define unique conditions that effect the operating sequence, or define a particular condition of an operating sequence. For instance:

1) FT - Repeat (flip-flop) when set denotes that the next instruction is to be repeated until prespecified terminate conditions have been met.

2) FD - Repeat Double (flip-flop) when set denotes that the next even/odd pair of instructions are to be repeated until prespecified terminate conditions have been met.

3) FL - Repeat Link (flip-flop) when set denotes that the next instruction is to be repeated using an operand link address contained within the data fetched from memory. The instruction will be repeated until a prespecified terminate condition has been met or until a "link address" of zero has been found.

4) RF - Repeat First (flip-flop) when set defines the first time that an effective address for an operand is prepared when in a Repeat, Repeat Double (first time for the pair of the repeated instructions), or Repeat Link mode of operation.

5) TRMF - Terminate Repeat (flip-flop) is set upon detection of a Repeat mode terminate condition. It enables the Control Frame to enter a Terminate mode of operation designated TERM. TERM and the repeat flags are reset at end of the current prepare cycle (PA if no indirects, PN if indirects).

6) XDE - Execute Double Even (flip-flop) is set if an execute double instruction occurs as an even instruction or if IC is reset when an execute double instruction is forced into the Y- and C-registers (for a fault or in response to an interrupt request).

9.2.1.4 - contd

7) XDO - Execute Double Odd (flip-flop) is set if an execute double instruction occurs as an odd instruction or if IC is set when an execute double instruction is forced into the Y- and C-registers (for a fault or in response to an interrupt request).

8) MSY - Master/Slave for Y (flip-flop) is set when instructions are loaded into the Y- and C-registers if the addressing mode is absolute or if the segment descriptor of the current procedure indicates Procedure-Master when the addressing mode is append. The descriptor of the current procedure is defined by bits 27-35 of the Segment Descriptor Word contained in the Descriptor Register of the Interface Frame. MSY is set whether the instructions to the Y- and C-registers are from the data-in bus (ZDI) or the I-register.

9) MSI - Master/Slave for I (flip-flop) is set whenever instructions are loaded into the I-register when the segment descriptor of the current procedure indicates Procedure-Master.

10) ABS - Absolute (flip-flop) when set indicates that addressing is to be done in the Absolute mode (refer to paragraph 2.3). When reset addressing is done in the Append mode except for fault and interrupt responses. This flip-flop corresponds to bit 28 in the indicator register.

11) MASF - Temporary Absolute (flip-flop) when set indicates that the Processor is addressing in the Absolute mode. This mode is entered when instruction pair that is "brute forced" as the result of a fault or interrupt (Processor is in a WF cycle and not stopped). The temporary absolute flip-flop may also be effected by an RCU instruction.

9.2.2 INSTRUCTION AND OPERAND ADDRESS MODES

As mentioned previously in paragraph 2.3, addressing in the GE-645 is done in either the Absolute or the Append mode as specified by the condition of the Absolute (ABS) flip-flop in the Control Frame. Addressing is normally done in the Append mode. In the Append mode the effective address in the BS-adder is appended with an address pointer from the Base Frame in the RS-adder of the Interface Frame to form a 18-bit address, which is then formulated into a 24-bit address for the core location.

When addressing for instructions in the Absolute mode, the effective address becomes the core location that is accessed. When addressing for operands in the Absolute mode however, the effective address may be appended to form a 24-bit address. This is possible if an address pointer is generated because bit 29 of the instruction word is a "1" or because an ITS or ITB modifier is received during any operand address preparation cycle. If appending is done for operands while in the Absolute mode, the state of the Absolute flip-flop is not effected, but a strobe which tells the Interface Frame to ignore any appending is inhibited.

The Absolute flip-flop is set by one of the following conditions.

1) Addressing is being done in a Temporary Absolute mode because of a fault or external interrupt condition and a satisfied transfer (other than a TSS), that does not involve generation of an address pointer for the transfer address is executed. An address pointer is generated if bit 29 of the transfer is a "1" or ITS or ITB modifier is involved in obtaining the transfer address.

2) A RTD, RET or RCU instruction fetched from a Procedure-Master segment, is executed and it causes bit 28 of the Indicator Register to set (this bit corresponds to the Absolute indicator).

9.2.2 - contd

The Append mode is entered (absolute flip-flop is reset) via one of the following:

1) A RTD, RET or RCU instruction, fetched from a Procedure-Master segment is executed and it causes bit 28 of Indicator Register to reset.

2) A TSS instruction is executed.

3) A satisfied transfer that generates an address pointer is executed (bit 29 of the transfer is a "1" or an ITS or ITB modifier is encountered in obtaining the transfer address).

Temporary Absolute mode (MASF - flip-flop set) is entered when a fault condition has been detected and is about to be serviced (if the Processor is not stopped), or if an RCU restores it to the set state. During the temporary Absolute state, a "brute force" XED instruction is formulated and the fault vectors are executed. The Temporary Absolute mode is exited to either the Absolute or the Append mode (providing another fault does not occur in the XED) if one of the fault vectors contains:

1) An RCU instruction which restores it to the reset state.

2) A RTD or RET instruction or any satisfied transfer.

If neither of the above two conditions are met MASF is reset when a PI cycle is entered after the XED is c completed.

9.2.2.1 Operand Address Modification

With few exceptions, all instructions in the GE-645 instruction set provide for some form of address modification for the operand. The different types of address modification are described in paragraph 2.6.1. A flow chart of the different types of address modification is presented in Figure 9.2.2.1. This flow chart is rather general in nature but contains sufficient

---

9.2.2.1 - contd

information to enable the reader to follow the address modification sequence in conjunction with the Control Frame block diagram, 43D160212 sheet 5. The detailed address modification sequences are covered in paragraph 9.4.

R (Register) Modification

With R modification (t_m of instruction word tag = 00) one of 13 registers are added to the instruction address (bits 0-17 of instruction contained in the Y- or C-register) in the YS-adder. If the register designator (t_d of instruction word) designates that bits 0-17 of the instruction word constitute a direct operand (DU, DL modification) then bits 0-17 of the instruction word are sent directly to the Register Frame of the Operations Unit via the BS-adder, and no memory access is required for the operand. However, if bit 29 of this instruction word is set, this indicates that bits 0-2 of the instruction word designate an address base register for address, appending and an internal base may be added to the address in the BS-adder before an operand is fetched. An R modification terminates any further address modification.

The operation code and tag portions (bits 18-35) of the instruction word are switched to the control point decode logic; the selection of the Y- or C-register for an instruction is determined by the reset or set condition of the IC flip-flop. For this discussion only the Y-register is considered (IC reset). Decoding of an R modifier in the t_m portion of the instruction word tag, allows the t_d portion to be switched to the ZX switch decode logic, via the ZI_{14-17} (ZI/ZT) switch, and select one of 13 registers for modification. The selected register is then added to the tenative operand address of the instruction word in the YS-adder. If t_d selects DU or DL modification, the operand address in the YS-adder is the operand. The output of the YS-adder is then present at the input to the BS-adder.

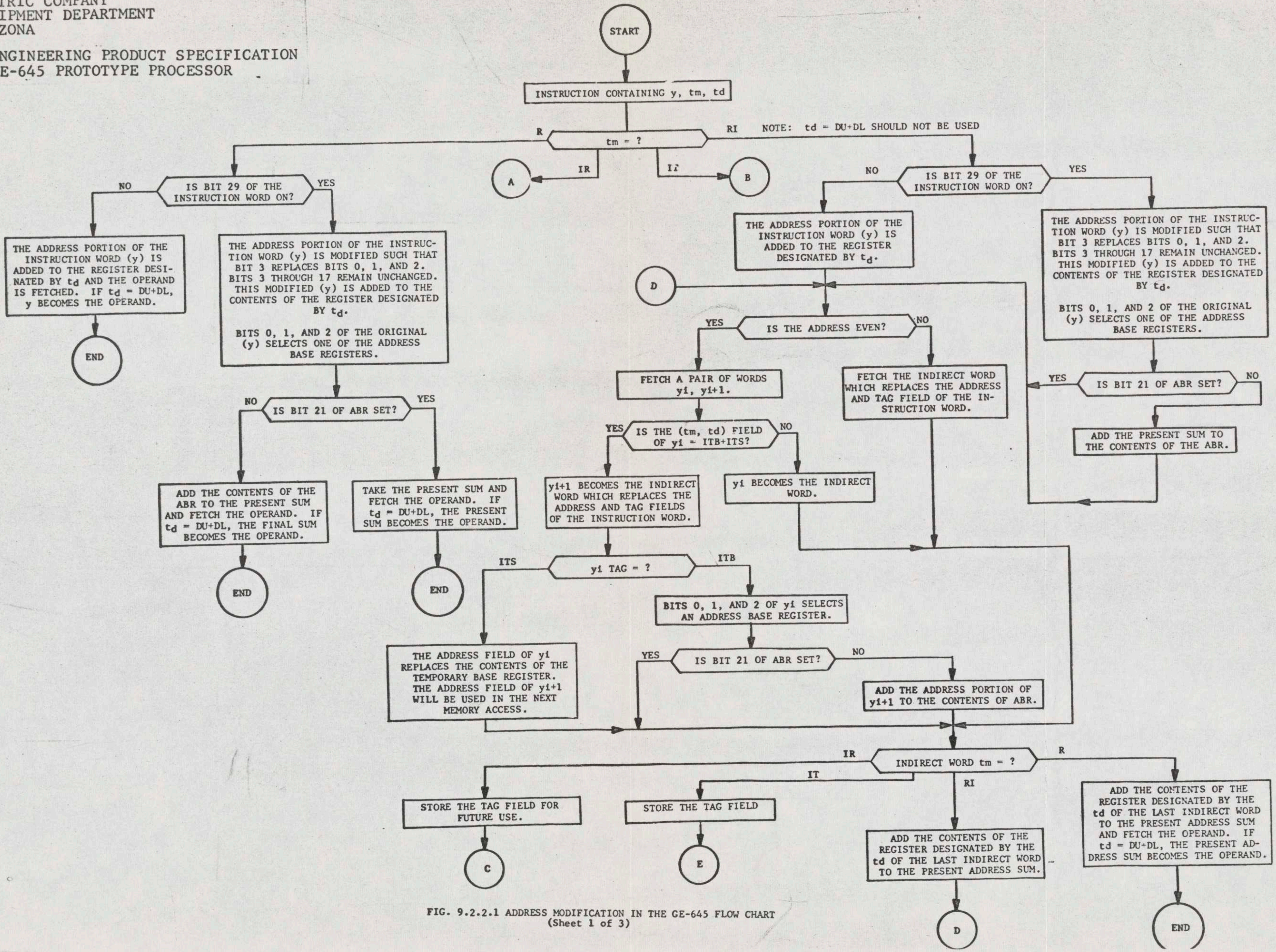

START
INSTRUCTION CONTAINING y, tm, td
tm = ?
R
RI
IR
IT
NOTE: td = DU+DL SHOULD NOT BE USED
A
B
IS BIT 29 OF THE INSTRUCTION WORD ON?
NO
YES
THE ADDRESS PORTION OF THE INSTRUCTION WORD (y) IS ADDED TO THE REGISTER DESIGNATED BY t_d AND THE OPERAND IS FETCHED. IF t_d = DU+DL, y BECOMES THE OPERAND.
END
THE ADDRESS PORTION OF THE INSTRUCTION WORD (y) IS MODIFIED SUCH THAT BIT 3 REPLACES BITS 0, 1, AND 2. BITS 3 THROUGH 17 REMAIN UNCHANGED. THIS MODIFIED (y) IS ADDED TO THE CONTENTS OF THE REGISTER DESIGNATED BY t_d.
BITS 0, 1, AND 2 OF THE ORIGINAL (y) SELECTS ONE OF THE ADDRESS BASE REGISTERS.
IS BIT 21 OF ABR SET?
NO
YES
ADD THE CONTENTS OF THE ABR TO THE PRESENT SUM AND FETCH THE OPERAND. IF t_d = DU+DL, THE FINAL SUM BECOMES THE OPERAND.
END
TAKE THE PRESENT SUM AND FETCH THE OPERAND. IF t_d = DU+DL, THE PRESENT SUM BECOMES THE OPERAND.
END
IS BIT 29 OF THE INSTRUCTION WORD ON?
NO
YES
THE ADDRESS PORTION OF THE INSTRUCTION WORD (y) IS ADDED TO THE REGISTER DESIGNATED BY t_d.
THE ADDRESS PORTION OF THE INSTRUCTION WORD (y) IS MODIFIED SUCH THAT BIT 3 REPLACES BITS 0, 1, AND 2. BITS 3 THROUGH 17 REMAIN UNCHANGED. THIS MODIFIED (y) IS ADDED TO THE CONTENTS OF THE REGISTER DESIGNATED BY t_d.
BITS 0, 1, AND 2 OF THE ORIGINAL (y) SELECTS ONE OF THE ADDRESS BASE REGISTERS.
D
IS THE ADDRESS EVEN?
YES
NO
FETCH A PAIR OF WORDS yi, yi+1.
FETCH THE INDIRECT WORD WHICH REPLACES THE ADDRESS AND TAG FIELD OF THE INSTRUCTION WORD.
YES
IS BIT 21 OF ABR SET?
NO
ADD THE PRESENT SUM TO THE CONTENTS OF THE ABR.
YES
IS THE (tm, td) FIELD OF yi = ITB+ITS?
NO
yi+1 BECOMES THE INDIRECT WORD WHICH REPLACES THE ADDRESS AND TAG FIELDS OF THE INSTRUCTION WORD.
yi BECOMES THE INDIRECT WORD.
yi TAG = ?
ITS
ITB
BITS 0, 1, AND 2 OF yi SELECTS AN ADDRESS BASE REGISTER.
THE ADDRESS FIELD OF yi REPLACES THE CONTENTS OF THE TEMPORARY BASE REGISTER. THE ADDRESS FIELD OF yi+1 WILL BE USED IN THE NEXT MEMORY ACCESS.
YES
IS BIT 21 OF ABR SET?
NO
ADD THE ADDRESS PORTION OF yi+1 TO THE CONTENTS OF ABR.
INDIRECT WORD tm = ?
IR
IT
RI
R
STORE THE TAG FIELD FOR FUTURE USE.
C
STORE THE TAG FIELD
E
ADD THE CONTENTS OF THE REGISTER DESIGNATED BY THE td OF THE LAST INDIRECT WORD TO THE PRESENT ADDRESS SUM.
D
ADD THE CONTENTS OF THE REGISTER DESIGNATED BY THE td OF THE LAST INDIRECT WORD TO THE PRESENT ADDRESS SUM AND FETCH THE OPERAND. IF td = DU+DL, THE PRESENT ADDRESS SUM BECOMES THE OPERAND.
END

FIG. 9.2.2.1 ADDRESS MODIFICATION IN THE GE-645 FLOW CHART
(Sheet 1 of 3)

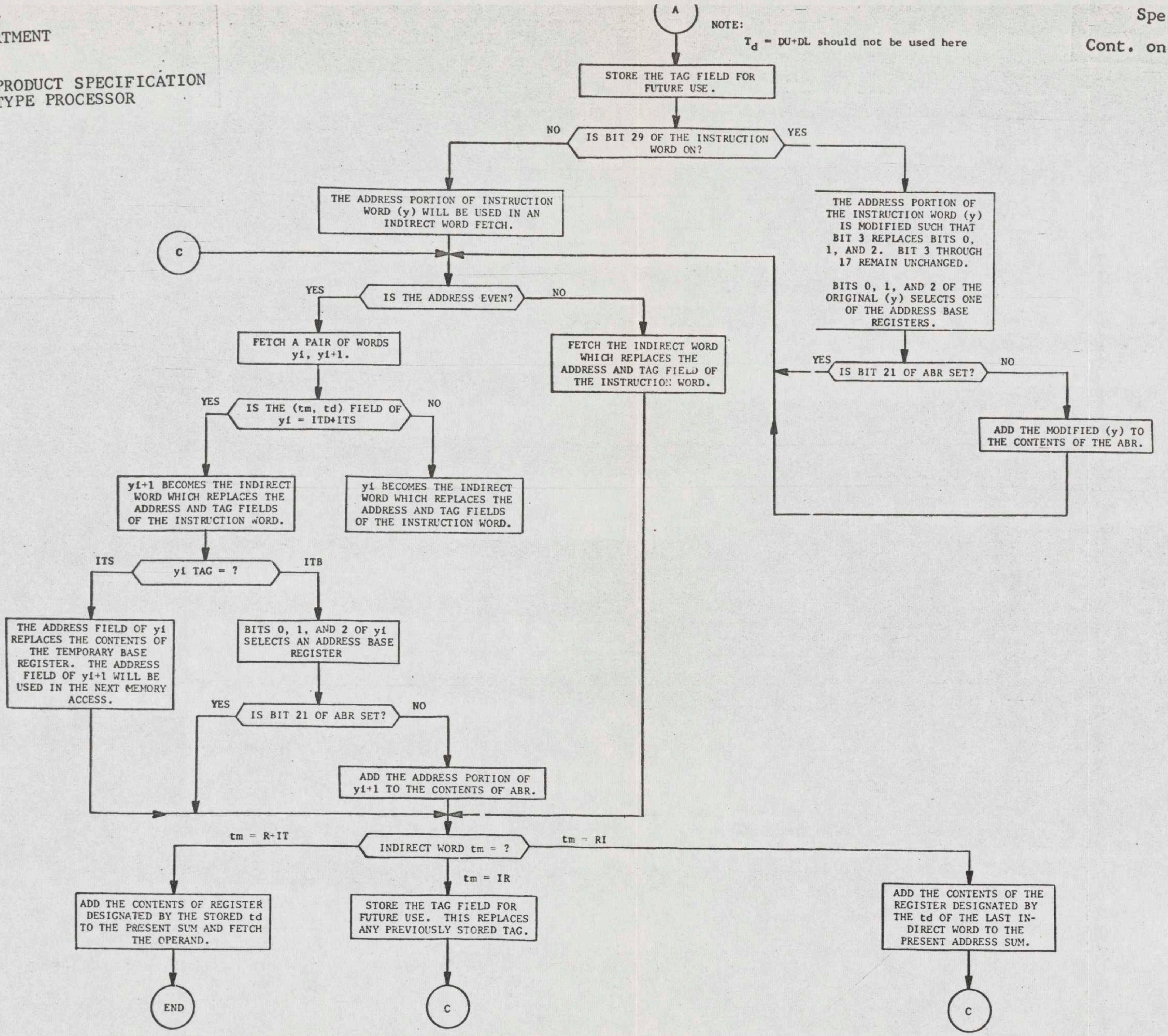

A
NOTE:
T_d = DU+DL should not be used here
STORE THE TAG FIELD FOR FUTURE USE.
NO
IS BIT 29 OF THE INSTRUCTION WORD ON?
YES
THE ADDRESS PORTION OF INSTRUCTION WORD (y) WILL BE USED IN AN INDIRECT WORD FETCH.
THE ADDRESS PORTION OF THE INSTRUCTION WORD (y) IS MODIFIED SUCH THAT BIT 3 REPLACES BITS 0, 1, AND 2. BIT 3 THROUGH 17 REMAIN UNCHANGED.
BITS 0, 1, AND 2 OF THE ORIGINAL (y) SELECTS ONE OF THE ADDRESS BASE REGISTERS.
C
YES
IS THE ADDRESS EVEN?
NO
FETCH A PAIR OF WORDS yi, yi+1.
FETCH THE INDIRECT WORD WHICH REPLACES THE ADDRESS AND TAG FIELD OF THE INSTRUCTION WORD.
YES
IS BIT 21 OF ABR SET?
NO
YES
IS THE (tm, td) FIELD OF yi = ITD+ITS
NO
ADD THE MODIFIED (y) TO THE CONTENTS OF THE ABR.
yi+1 BECOMES THE INDIRECT WORD WHICH REPLACES THE ADDRESS AND TAG FIELDS OF THE INSTRUCTION WORD.
yi BECOMES THE INDIRECT WORD WHICH REPLACES THE ADDRESS AND TAG FIELDS OF THE INSTRUCTION WORD.
ITS
yi TAG = ?
ITB
THE ADDRESS FIELD OF yi REPLACES THE CONTENTS OF THE TEMPORARY BASE REGISTER. THE ADDRESS FIELD OF yi+1 WILL BE USED IN THE NEXT MEMORY ACCESS.
BITS 0, 1, AND 2 OF yi SELECTS AN ADDRESS BASE REGISTER
YES
IS BIT 21 OF ABR SET?
NO
ADD THE ADDRESS PORTION OF yi+1 TO THE CONTENTS OF ABR.
tm = R+IT
INDIRECT WORD tm = ?
tm = RI
tm = IR
ADD THE CONTENTS OF REGISTER DESIGNATED BY THE STORED td TO THE PRESENT SUM AND FETCH THE OPERAND.
STORE THE TAG FIELD FOR FUTURE USE. THIS REPLACES ANY PREVIOUSLY STORED TAG.
ADD THE CONTENTS OF THE REGISTER DESIGNATED BY THE td OF THE LAST INDIRECT WORD TO THE PRESENT ADDRESS SUM.
END
C
C

FIG. 9.2.2.1 ADDRESS MODIFICATION IN THE GE-645 FLOWCHART
(Sheet 2 of 3)

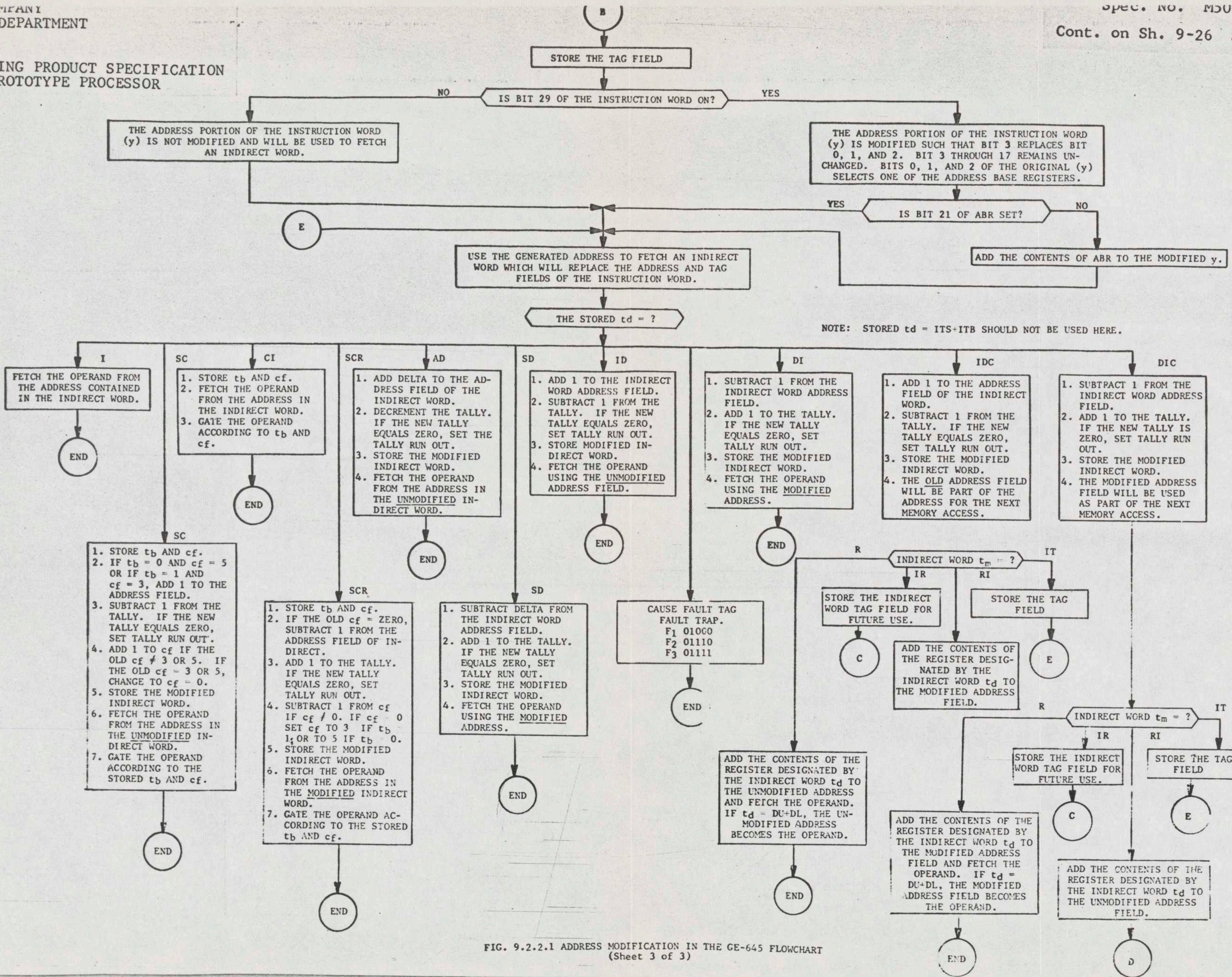

B
STORE THE TAG FIELD
IS BIT 29 OF THE INSTRUCTION WORD ON?
NO
YES
THE ADDRESS PORTION OF THE INSTRUCTION WORD (y) IS NOT MODIFIED AND WILL BE USED TO FETCH AN INDIRECT WORD.
THE ADDRESS PORTION OF THE INSTRUCTION WORD (y) IS MODIFIED SUCH THAT BIT 3 REPLACES BIT 0, 1, AND 2. BIT 3 THROUGH 17 REMAINS UNCHANGED. BITS 0, 1, AND 2 OF THE ORIGINAL (y) SELECTS ONE OF THE ADDRESS BASE REGISTERS.
YES
IS BIT 21 OF ABR SET?
NO
E
ADD THE CONTENTS OF ABR TO THE MODIFIED y.
USE THE GENERATED ADDRESS TO FETCH AN INDIRECT WORD WHICH WILL REPLACE THE ADDRESS AND TAG FIELDS OF THE INSTRUCTION WORD.
THE STORED td = ?
NOTE: STORED td = ITS+ITB SHOULD NOT BE USED HERE.
I
FETCH THE OPERAND FROM THE ADDRESS CONTAINED IN THE INDIRECT WORD.
END
SC
CI
1. STORE t_b AND c_f. 2. FETCH THE OPERAND FROM THE ADDRESS IN THE INDIRECT WORD. 3. GATE THE OPERAND ACCORDING TO t_b AND c_f.
END
SCR
AD
1. ADD DELTA TO THE ADDRESS FIELD OF THE INDIRECT WORD. 2. DECREMENT THE TALLY. IF THE NEW TALLY EQUALS ZERO, SET THE TALLY RUN OUT. 3. STORE THE MODIFIED INDIRECT WORD. 4. FETCH THE OPERAND FROM THE ADDRESS IN THE UNMODIFIED INDIRECT WORD.
END
SD
ID
1. ADD 1 TO THE INDIRECT WORD ADDRESS FIELD. 2. SUBTRACT 1 FROM THE TALLY. IF THE NEW TALLY EQUALS ZERO, SET TALLY RUN OUT. 3. STORE MODIFIED INDIRECT WORD. 4. FETCH THE OPERAND USING THE UNMODIFIED ADDRESS FIELD.
END
DI
1. SUBTRACT 1 FROM THE INDIRECT WORD ADDRESS FIELD. 2. ADD 1 TO THE TALLY. IF THE NEW TALLY EQUALS ZERO, SET TALLY RUN OUT. 3. STORE THE MODIFIED INDIRECT WORD. 4. FETCH THE OPERAND USING THE MODIFIED ADDRESS.
END
IDC
1. ADD 1 TO THE ADDRESS FIELD OF THE INDIRECT WORD. 2. SUBTRACT 1 FROM THE TALLY. IF THE NEW TALLY EQUALS ZERO, SET TALLY RUN OUT. 3. STORE THE MODIFIED INDIRECT WORD. 4. THE OLD ADDRESS FIELD WILL BE PART OF THE ADDRESS FOR THE NEXT MEMORY ACCESS.
DIC
1. SUBTRACT 1 FROM THE INDIRECT WORD ADDRESS FIELD. 2. ADD 1 TO THE TALLY. IF THE NEW TALLY IS ZERO, SET TALLY RUN OUT. 3. STORE THE MODIFIED INDIRECT WORD. 4. THE MODIFIED ADDRESS FIELD WILL BE USED AS PART OF THE NEXT MEMORY ACCESS.
SC
1. STORE t_b AND c_f. 2. IF $t_b = 0$ AND $c_f = 5$ OR IF $t_b = 1$ AND $c_f = 3$, ADD 1 TO THE ADDRESS FIELD. 3. SUBTRACT 1 FROM THE TALLY. IF THE NEW TALLY EQUALS ZERO, SET TALLY RUN OUT. 4. ADD 1 TO c_f IF THE OLD $c_f \neq 3$ OR 5. IF THE OLD $c_f = 3$ OR 5, CHANGE TO $c_f = 0$. 5. STORE THE MODIFIED INDIRECT WORD. 6. FETCH THE OPERAND FROM THE ADDRESS IN THE UNMODIFIED INDIRECT WORD. 7. GATE THE OPERAND ACCORDING TO THE STORED t_b AND c_f.
END
SCR
1. STORE t_b AND c_f. 2. IF THE OLD c_f = ZERO, SUBTRACT 1 FROM THE ADDRESS FIELD OF INDIRECT. 3. ADD 1 TO THE TALLY. IF THE NEW TALLY EQUALS ZERO, SET TALLY RUN OUT. 4. SUBTRACT 1 FROM c_f IF $c_f \neq 0$. IF $c_f = 0$ SET c_f TO 3 IF $t_b = 1$; OR TO 5 IF $t_b = 0$. 5. STORE THE MODIFIED INDIRECT WORD. 6. FETCH THE OPERAND FROM THE ADDRESS IN THE MODIFIED INDIRECT WORD. 7. GATE THE OPERAND ACCORDING TO THE STORED t_b AND c_f.
END
SD
1. SUBTRACT DELTA FROM THE INDIRECT WORD ADDRESS FIELD. 2. ADD 1 TO THE TALLY. IF THE NEW TALLY EQUALS ZERO, SET TALLY RUN OUT. 3. STORE THE MODIFIED INDIRECT WORD. 4. FETCH THE OPERAND USING THE MODIFIED ADDRESS.
END
CAUSE FAULT TAG FAULT TRAP. F_1 01000 F_2 01110 F_3 01111
END
R
INDIRECT WORD t_m = ?
IT
IR
RI
STORE THE INDIRECT WORD TAG FIELD FOR FUTURE USE.
STORE THE TAG FIELD
C
E
ADD THE CONTENTS OF THE REGISTER DESIGNATED BY THE INDIRECT WORD t_d TO THE MODIFIED ADDRESS FIELD.
ADD THE CONTENTS OF THE REGISTER DESIGNATED BY THE INDIRECT WORD t_d TO THE UNMODIFIED ADDRESS AND FETCH THE OPERAND. IF t_d = DU+DL, THE UNMODIFIED ADDRESS BECOMES THE OPERAND.
END
R
INDIRECT WORD t_m = ?
IT
IR
RI
STORE THE INDIRECT WORD TAG FIELD FOR FUTURE USE.
STORE THE TAG FIELD
C
E
ADD THE CONTENTS OF THE REGISTER DESIGNATED BY THE INDIRECT WORD t_d TO THE MODIFIED ADDRESS FIELD AND FETCH THE OPERAND. IF t_d = DU+DL, THE MODIFIED ADDRESS FIELD BECOMES THE OPERAND.
ADD THE CONTENTS OF THE REGISTER DESIGNATED BY THE INDIRECT WORD t_d TO THE UNMODIFIED ADDRESS FIELD.
END
D

FIG. 9.2.2.1 ADDRESS MODIFICATION IN THE GE-645 FLOWCHART
(Sheet 3 of 3)

9.2.2.1 - contd

If bit 29 of the instruction word is "1," and an internal base is designated by bits 0-2 of the instruction word, then the address field of the internal ABR is added to the address in the BS-adder during the PA prepare address cycle for an operand. Whenever an ABR is designated in an instruction word (bit 29 equals a "1"), bit 3 of the instruction word is extended to fill bits 0-2. This is done at the input to the YS-adder.

RI (Register then Indirect) Modification

For RI modification (t_m = 01), the register specified by t_d of the instruction word tag is used to modify the operand address in the same manner as previously discussed for the R modification. However, after adding the specified register and internal base register if specified, the effective address is then used to fetch an "indirect word" (a new word from memory that is the result of address modification). This "indirect word" replaces portions of the original instruction word in the Y- or C- register as follows:

- bits 0-17 replace the address
- bits 18-29 are ignored (op code of original remains)
- bits 30-35 replace the tag (t_m, t_d)

Any address modification called for by this indirect word is then executed. If an R modifier is called for in the indirect word, then the indirection terminates after the designated register is added. The effective address is then used to fetch the operand. Naturally, the appending of the various ABR's, the PBR etc. in the Interface Frame to form the memory address is performed as necessary for all memory accesses unless the Processor is in Absolute mode.

When any modifier is found in the t_m field of the new tag brought in by the indirect word, the indirection process continues as specified for an RI, IR or IT modification, until an R modifier is found which ends the chain. Also, whenever IR or RI modification is specified and the effective address in the BS-adder designates an even location, two indirect words are

9.2.2.1 - contd

pulled in anticipation of an ITS or ITB modifier in the even indirect word. The first indirect word is strobed into the working register (Y or C) in the normal manner, and if it specifies ITS or ITB modification, then the odd indirect word of the pair gets strobed on top of the first (even) indirect word. If no ITS or ITB modification is detected, then strobing of the second (odd) indirect word is inhibited.

IR (Indirect then Register) Modification

When performing indirect then register modification ($t_m = 11$) the operand address, from the instruction word now present in the BS-adder, is used to fetch an indirect word. An internal base may be added to the instruction word address field prior to the first indirect word fetch if so specified by bits 0-2 of the instruction word address field. Any modification to be performed is as indicated by the t_m, t_d portion of the original word, which is stored in the CT-register just prior to the time that an indirect word is fetched from memory. When the indirect word is brought in from memory, it is strobed into the Y- or C-register depending on whether the even or odd instruction is being processed. If the t_m field of the indirect word has an R or IT modifier, then the t_d field of the original word that was stored in the CT-register is used to select the register, via the CT-switch and ZX decoder, which modifies the indirect word address. A second indirect word is then fetched from memory and strobed into the Y- or C-register. Thereafter, operation is as specified for RI modification. When an IR modifier is detected in the indirect word resulting from an IR modifier in the instruction word, then the new t_m, t_d field replaces the old t_m,t_d field in the CT-register. Like RI modification, if the indirect word is in an even location, an even/odd pair of words are pulled in anticipation of an ITS or ITB tag in the even word of the pair.

9.2.2.1 - contd

IT modification ($t_m = 10$) involves using the operand address in the instruction word to fetch an indirect then tally word (except for modifications F1, F2, F3), then operating on the location specified by the indirect word according to the operation code and tally designator (t_d) of the original word. For IT modification, any one of 13 variations specified by the t_d field are possible. The ITS and ITB modifications involve both the t_m,t_d field of indirect words pulled as a result of RI or IR modification. These can really be considered an Indirect Pair modification and do not fall in any one of the four defined types (R, RI, IR, IT) even though the t_m, t_d fields fall in the IT modification code range.

Indirect then tally words fall into four general categories depending on how the tag (t_m;t_d) of the instruction word, or indirect then tally word is interpreted:

1) I, DI, ID, F1, F2, F3 -- these tally designators are ignored in the indirect then tally word but are specified by the t_d field of the instruction word.

2) DIC, IDC -- these designators in the instruction word cause the t_m,t_d field of the indirect then tally word to be interpreted as a normal t_m,t_d and may designate R, RI, IR, or IT modification.

3) CI, SC, SCR -- these designators in the instruction word cause the t_m, t_d field of the indirect then tally word to be interpreted as a character size and character position for the operand specified by the address portion of the indirect then tally word.

4) AD, SD -- the tag field of the indirect then tally word is interpreted as a delta which is to be added or subtracted to the address of the operand.

9.2.2.1 - contd

The tally portion of the indirect then tally word is contained and updated in the Interface Frame. The tag portion is ignored, is strobed into the CT-register, is sent to the Operations Unit and Interface Frame, or is added to the operand address in the YS-adder, dependent on the specific tally designator and the operations involved.

1) F1 (0000), F2 (0110), F3 (0111) -- detection of these designators in the instruction word causes an immediate fault sequence to be initiated in the Processor. This results in a "brute forced" XED instruction in the Y- and C-registers which generates a fault vector address for the specific fault tag; F1, F2 or F3. These fault tags all cause the Control Frame to respond in the same manner, the only difference being in the fault vector address associated with each tag.

2) I (1001) -- This designator results in the fetching of an indirect word using the address in the instruction word that is also present in the BS-adder. The resultant indirect-word address is strobed into the address portion of the Y- or C-register and the tag portion is ignored. Fetching of the operand is accomplished using the address portion of the indirect word now present in the BS-adder. (The YS- and BS-adders add whatever is presented to their inputs).

3) DI (1100) -- When a DI modifier is detected, the address in the BS-adder is used to fetch an indirect word which consists of an address (bits 0-17), a tally (bits 18-29), and a tag. The address portion is strobed into the Y- or C-register, depending on the location of the original instruction. The tally portion is strobed into the RS-adder in the Interface Frame, and the tag portion is ignored. The new operand address is decremented by "1" and the tally portion is incremented by "1." The resultant new address and tally are rewritten back into the original memory location of the indirect word, then the operand is fetched with the address

---

9.2.2.1 - contd

obtained from the modified indirect word (original address minus "1") now present in the BS-adder. Decrementing of the operand address takes place in the YS-adder, and the decode of the DI modifier is taken from the CT-register outputs. The CT-register receives the instruction word tag just prior to the indirect cycle for the indirect then tally word. Every time that the indirect then tally location is accessed by the instruction word, the address of the indirect then tally word is decremented by "1" and the tally field is incremented by "1." When the tally reaches zero the Tally Runout Indicator is turned on.

4) ID (1110) -- This modifier is essentially the inverse of the DI modifier. The operand address in the instruction word is used to fetch an indirect then tally word in the same manner as for DI modification. However, the address field is incremented by "1" and the tally field decremented by "1" before rewriting it in the memory location from which it was obtained. Then the operand is fetched using the original address of the indirect then tally word that is now present in the BS-adder from the Y- or C-register. Incrementing of the operand address contained in the indirect then tally word is done in the YS-adder. As soon as the modified address is rewritten back into memory the original address is restored. This is because "1" is removed from the YS-adder input and only the indirect then tally word address is still present. When the tally field reaches zero the Tally Runout Indicator is turned on.

5) DIC (1101) -- The DIC modifier is the same as the DI modifier with the following exception: the tag portion of the indirect then tally word brought into the Y-or C-register is treated as a normal tag, after the address and tally portions of the indirect word are updated and rewritten into the memory location from which it was obtained. The new tag is strobed into the CT-register and may specify R, RI, IR, or IT modification (R must = N); in which case modification is performed as specified.

9.2.2.1 - contd

If no further indirect word was specified by the tag, then the operand is fetched using the address or the modified indirect word (original address minus "1").

6) IDC (1111) -- This modifier is the same as the ID modifier with the following exception: the tag portion of the indirect then tally word brought into the Y- or C-register is treated as a normal tag after the address and tally portions of the indirect word are updated, and rewritten into the memory location from which it was obtained. The new tag is strobed into the CT-register and may specify R, RI, IR or IT modification (R must = N); in which case the modification is performed as specified. If no further indirect word was specified by the new tag, then the operand is fetched using the address of the original indirect word still located in the Y- or C- register, and now present in the BS-adder. When the tally portion of the indirect word reaches zero, then the Tally Runout Indicator is turned on.

7) ITS (100011) and ITB (100001) -- These modifiers are different than the normal IT modification since they are only recognized as a result of RI or IR address modification specified by the tag field of an instruction or indirect word that designates an even address. With this type of address modification, two words are pulled from memory, in a double-precision read-restore operation. The first word is unconditionally strobed into the Y- or C-register. Then the tag portion is decoded from the ZI_{12-17} lines for a possible ITS of ITB tag. If these tags are detected, the Control Frame stops any further address preparation until the second word of the pair is received from the memory. The second indirect word is a normal indirect word and is strobed on top of the first word that contained the ITS or ITB tag. Operand address preparation is now resumed in the normal manner, but with the new indirect word that is relative to a new address base. For ITS modification, the new base is strobed

9.2.2.1 - contd

into the TBR in the Base Frame; for ITB modification the new base is referenced from the segment tag registers in the Base Frame which now contain a new segment tag brought in by the ITB modification. The new base specified by the segment tag register in the Base Frame is then added in the BS-adder in the Control Frame if the ITB tag specified an internal base.

In summary: ITS and ITB modification is a method of changing the address base or "pointer" to a different address base; whether it be to a new segment or a different base within a segment. The Control Frame receives the first indirect word in the conventional manner, decodes the tag from the Y- or C-register, and if ITS or ITB modification, holds up operand address preparation until the second indirect word is brought in on top of the first. When no ITS or ITB tag is detected the second indirect word is ignored.

8) CI (1000) -- With this modifier, an indirect then tally word is obtained from the address contained in the instruction word. The indirect word address field replaces the operand address of the instruction word in the Y- or C-register, and the tag portion goes to the Operations Unit or the Interface Frame, depending on whether the operation to be performed is a Load or Store operation. The t_m field of the tag of the indirect then tally word defines a 6-bit or 9-bit character to be operated upon, and the t_d field defines the particular character involved. Fetching of the operand character is accomplished by using the operand address now present in the BS-adder from the Y- or C-register.

9) SC (1010) -- This modifier deals with 6-bit and 9-bit operand character sequencing. The instruction word address is used to fetch an indirect then tally word consisting of an operand address, tally, and tag field respectively. Only the operand address is strobed into the Y- or C-register. For

9.2.2.1 - contd

SC modification, the tag field of the indirect then tally word denotes the particular character or byte to be operated upon. Whenever this indirect then tally word is referenced, the byte position is increased by one, the tally is decreased by one, and it is rewritten back into the memory location from which it was obtained. When the tally field reaches zero, the Tally Runout Indicator is turned on. The byte position will progress from zero, to three or five and then start over from zero.

When the byte position goes from five to zero (6-bit bytes) or three to zero (9-bit bytes) the address of the operand is increased by one to obtain the memory location of the next operand. This is accomplished by decoding the byte position from bits 33-35 on the ZDI lines as the indirect then tally words are brought in from memory, and adding +1 to the address in the YS-adder when the byte count reaches maximum (5 or 3 indicating 6 or 4 bytes). Operation on the byte is done using the original byte specified by the first indirect word.

10) SCR (0101) -- SCR modification is the inverse of SC modification. The indirect then tally word is brought into the Y- or C-register in the same manner. But for SCR modification, whenever the indirect word is referenced, the byte position is decreased by one, the tally field is increased by one, and the new information is written back into the location from which it was obtained. Whenever the tally field reaches zero, the Tally Runout Indicator is turned on. Operation on the byte position is done using the new character position and new address. As before, the tag of the indirect then tally word is decoded from the ZDI lines as the words are brought from memory. Whenever zero is detected, -1 is added to the address in the YS-adder, and the tag field indicating the byte position is set to maximum (5 or 3 for 6 or 4 bytes).

9.2.2.1 - contd

11) AD (1011) -- With this type of address modification the instruction address is used to fetch an indirect then tally word containing an operand address, tally, and delta field; delta occupying bits 30-35 of the word. The operand address is strobed into the Y- or C-register, and the delta field is routed through the V portion of the ZX switch to be added to the operand address. The tally is decreased by "1." When the tally field reaches zero, the Tally Runout Indicator is turned on. The operand is then fetched using the original address contained in the indirect then tally word that is still present in the Y- or C-register.

12) SD (0100) -- This modification is the inverse of AD. Each time the indirect location is accessed, the operand address is decreased by delta, the tally is incremented by one, and they are rewritten back into the location from where they were obtained. When the tally reaches zero the Tally Runout Indicator is turned on. The operand is now fetched using the modified address still present in the YS-adder corresponding to the updated indirect then tally word. Subtracting delta is done through the $\overline{V}$ and -I portion of the ZX switch which have as their inputs the delta portion of the indirect word from the DV 00/35 OB (ZDI) data input lines and logic 1's. To subtract, the 2's complement of delta must be added in the YS-adder.

9.2.3 INSTRUCTION EXECUTION MODES

The modes of execution for the Processor which were described in paragraph 2.3.3, are defined by two flip-flops in the Control Frame designated MSY and MSI. These flip-flops indicate that the instructions fetched from memory, and now occupying the Y-, C-, and I-registers, were fetched from a Procedure-Master segment; or that the Processor is in the Temporary Absolute or Absolute mode of address. Information concerning segment types is sent from the Interface Frame and is defined by the descriptor field of a Segment Descriptor Word or Page Table Word in the descriptor register.

9.2.3.1 MSI (Master in I-Register)

In the Append mode the MSI flip-flop being set indicates that the instructions contained in the I-register were fetched from a Procedure-Master segment. This flip-flop is normally reset since the majority of segments available to a "user" are not Procedure-Master. However, whenever instructions are being fetched to fill the I-register, the strobe that strobes the even word of the pair into the I-register, bits 0-35, also strobes the MSI flip-flop. If the descriptor field of the SDW or PTW contained in the descriptor register in the Interface Frame indicates Procedure-Master, MSI is set; otherwise it is reset. Whenever an RCU instruction is being executed, MSI may be changed to reflect the stored RCU data.

MSY (Master in Y- and C-Register)

In the Append mode this flip-flop indicates that the instructions now in the Y- and C-registers that were in the I-register or were brought in from the ZDI lines, were fetched from a Procedure-Master segment. Whenever the Processor is in the Temporary Absolute or Absolute mode of address, the Execution mode is forced to master. If the instruction fetch to fill the Y- and C-registers is the result of an XED instruction (which is not the result of a fault or an external interrupt) then the MSY flip-flop may not change until the XEDed instructions are executed.

9.2.4 INSTRUCTION FETCHES

The fetching of instructions for the GE-645 Prototype Processor is initiated and controlled by the Control Frame. Since the normal condition of the Control Frame is to keep four instructions in the instruction registers at all times, the sequences become very involved in the cases of instructions causing a transfer of control (faults all transfers, plus the RET, RTD, RCU, XEC and XED instructions). In addition, to pick up additional speed in cases where there is a load-store instruction combination in the I-register, the instruction fetch for reloading the I-register is postponed until after the operand address preparation cycle of the "load" instruction. There are many other pecularities which can cause the instruction fetch sequence to be altered slightly. These sequences will be covered in detail in paragraph 9.4. A flow chart of the "normal" instruction fetch sequence is shown in Fig. 9.2.4. This flow chart shows a normal two-instruction, or four-instruction fetch with the possibility of a load-store instruction combination, or transfer instruction in the I-register, and the possibility of an XED instruction in the Y- or C-registers.

Referencing both Fig. 9.2.4 and 43D160212 sheet 5, assume that the Processor has been initialized in some manner (discussed in the section or fault logic) and starts to fetch instructions. Normally starting out will require a four-instruction fetch (two at a time). The Control Frame enters a PI cycle for the instructions, and the contents of the ICTB-register are switched to the YS-adder input. The resultant address to fetch the instruction pair is the address from the BS-adder which is sent to the Interface Frame to fetch the instruction pair. Address appending may or may not take place in the Interface Frame, depending on the state of the Control Frame's absolute flip-flop. When the instruction address is sent to the Interface Frame, the Control Frame enters into a WI_2 cycle to wait for the instruction, ICTB is moved into ICTC, and +2 is added to ICTB (+1 added in both the YS- and BS-adders). Instructions are normally fetched in pairs and when the instructions come from memory, they are loaded into the I-register.

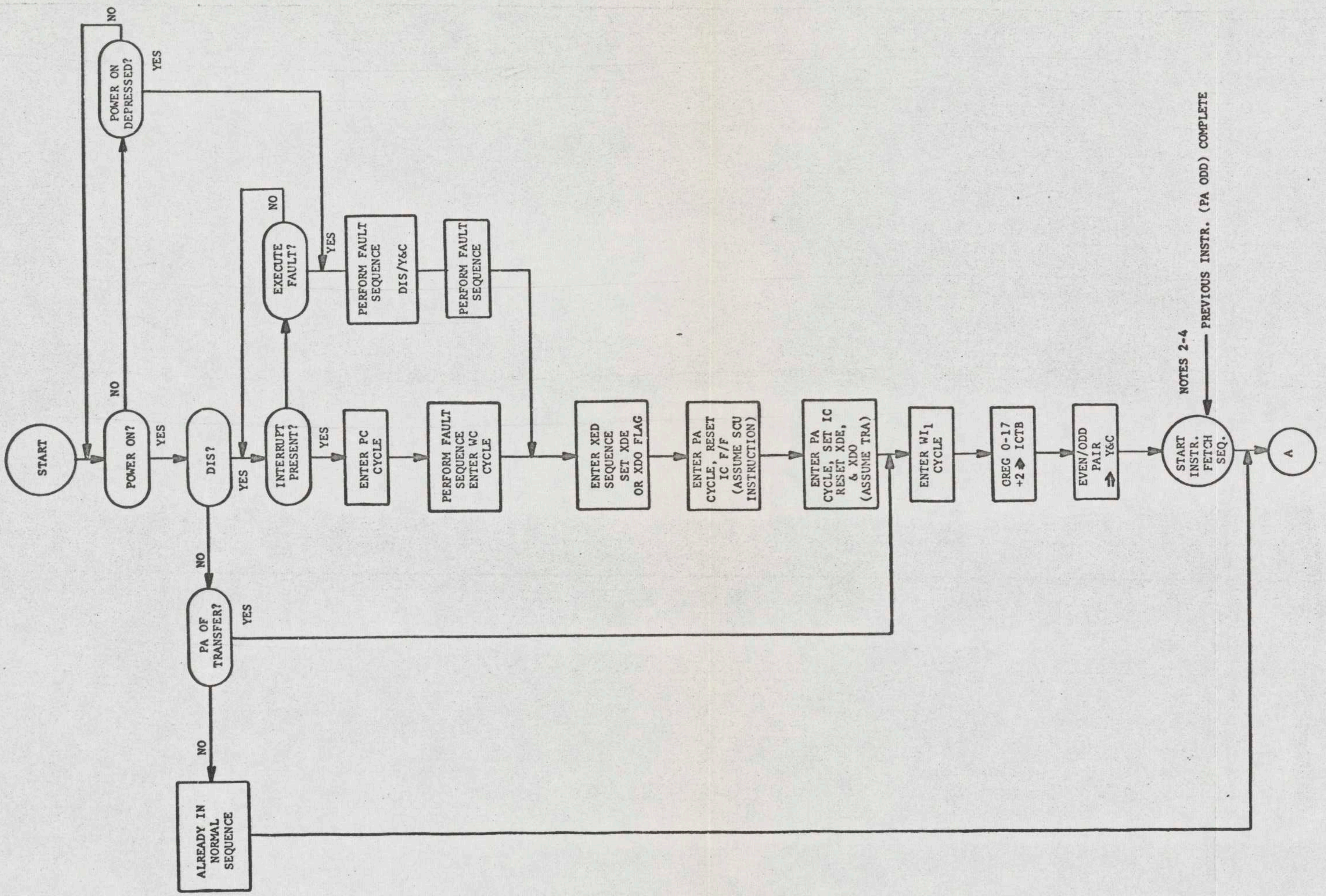
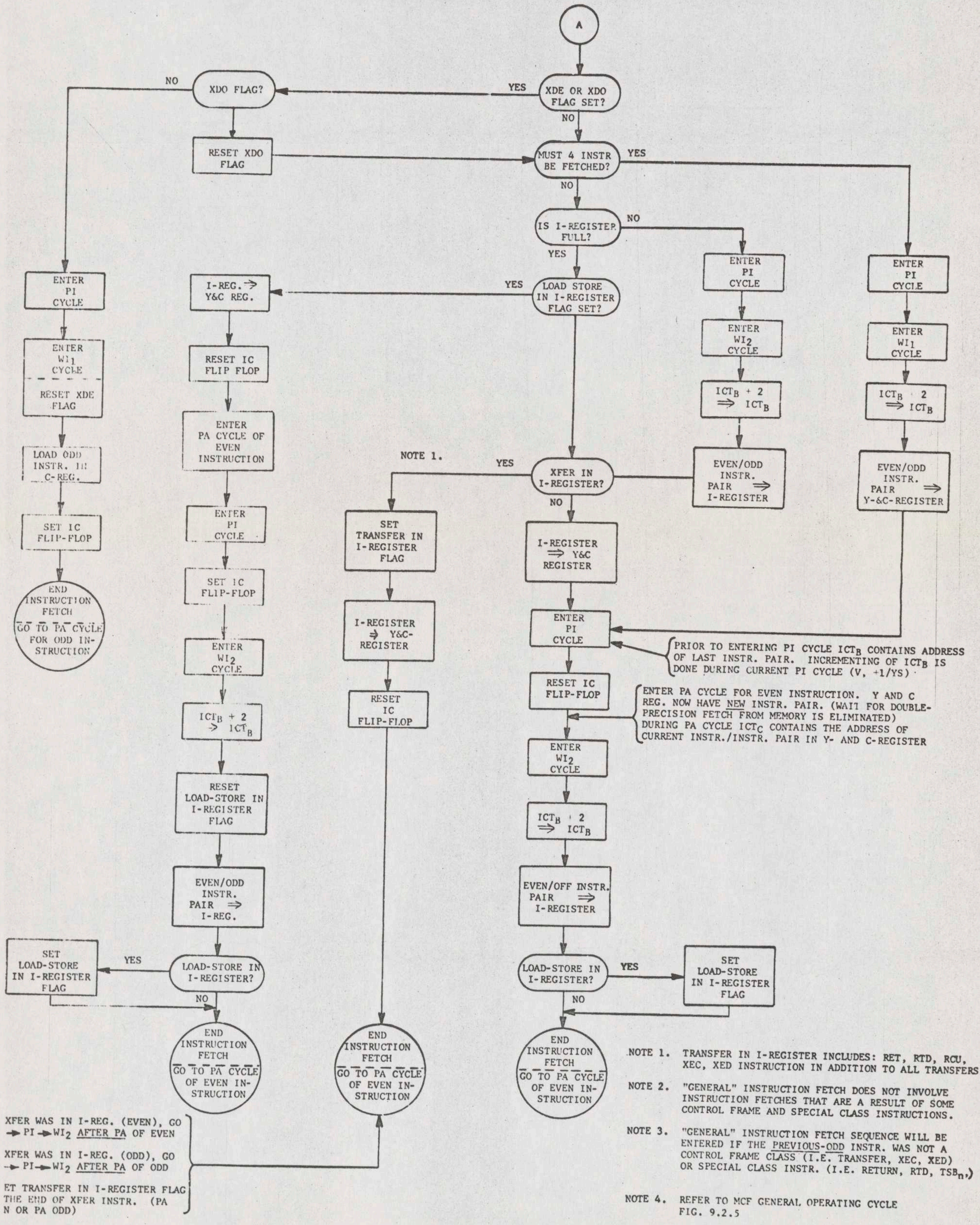

START
POWER ON?
NO
YES
POWER ON DEPRESSED?
NO
YES
DIS?
YES
NO
INTERRUPT PRESENT?
YES
EXECUTE FAULT?
NO
YES
PERFORM FAULT SEQUENCE DIS/Y&C
PERFORM FAULT SEQUENCE
ENTER PC CYCLE
PERFORM FAULT SEQUENCE ENTER WC CYCLE
PA OF TRANSFER?
YES
NO
ALREADY IN NORMAL SEQUENCE
ENTER XED SEQUENCE SET XDE OR XDO FLAG
ENTER PA CYCLE, RESET IC F/F (ASSUME SCU INSTRUCTION)
ENTER PA CYCLE, SET IC RESET XDE, & XDO (ASSUME TRA)
ENTER WI$_1$ CYCLE
GREG 0-17 +2 ➡ ICTB
EVEN/ODD PAIR ➡ Y&C
NOTES 2-4
PREVIOUS INSTR. (PA ODD) COMPLETE
START INSTR. FETCH SEQ.
A

FIG. 9.2.4 GENERAL INSTRUCTION FETCH SEQUENCE,
FLOWCHART (sheet 1 of 2)


A
XDE OR XDO FLAG SET?
YES
NO
XDO FLAG?
NO
RESET XDO FLAG
MUST 4 INSTR BE FETCHED?
YES
NO
IS I-REGISTER FULL?
NO
YES
LOAD STORE IN I-REGISTER FLAG SET?
YES
I-REG. ⇒ Y&C REG.
RESET IC FLIP FLOP
ENTER PA CYCLE OF EVEN INSTRUCTION
ENTER PI CYCLE
SET IC FLIP-FLOP
ENTER WI2 CYCLE
ICTB + 2 → ICTB
RESET LOAD-STORE IN I-REGISTER FLAG
EVEN/ODD INSTR. PAIR ⇒ I-REG.
LOAD-STORE IN I-REGISTER?
YES
SET LOAD-STORE IN I-REGISTER FLAG
NO
END INSTRUCTION FETCH
GO TO PA CYCLE OF EVEN INSTRUCTION
ENTER PI CYCLE
ENTER WI1 CYCLE
RESET XDE FLAG
LOAD ODD INSTR. IN C-REG.
SET IC FLIP-FLOP
END INSTRUCTION FETCH
GO TO PA CYCLE FOR ODD INSTRUCTION
ENTER PI CYCLE
ENTER WI2 CYCLE
ICTB + 2 ⇒ ICTB
EVEN/ODD INSTR. PAIR ⇒ I-REGISTER
ENTER PI CYCLE
ENTER WI1 CYCLE
ICTB 2 ⇒ ICTB
EVEN/ODD INSTR. PAIR ⇒ Y-&C-REGISTER
NOTE 1.
YES
XFER IN I-REGISTER?
NO
SET TRANSFER IN I-REGISTER FLAG
I-REGISTER ⇒ Y&C-REGISTER
RESET IC FLIP-FLOP
END INSTRUCTION FETCH
GO TO PA CYCLE OF EVEN INSTRUCTION
I-REGISTER ⇒ Y&C REGISTER
ENTER PI CYCLE
RESET IC FLIP-FLOP
ENTER WI2 CYCLE
ICTB + 2 ⇒ ICTB
EVEN/OFF INSTR. PAIR I-REGISTER
LOAD-STORE IN I-REGISTER?
YES
SET LOAD-STORE IN I-REGISTER FLAG
NO
END INSTRUCTION FETCH
GO TO PA CYCLE OF EVEN INSTRUCTION

PRIOR TO ENTERING PI CYCLE ICT_B CONTAINS ADDRESS OF LAST INSTR. PAIR. INCREMENTING OF ICT_B IS DONE DURING CURRENT PI CYCLE (V, +1/YS)

ENTER PA CYCLE FOR EVEN INSTRUCTION. Y AND C REG. NOW HAVE NEW INSTR. PAIR. (WAIT FOR DOUBLE-PRECISION FETCH FROM MEMORY IS ELIMINATED) DURING PA CYCLE ICT_C CONTAINS THE ADDRESS OF CURRENT INSTR./INSTR. PAIR IN Y- AND C-REGISTER

XFER WAS IN I-REG. (EVEN), GO → PI → WI_2 AFTER PA OF EVEN

XFER WAS IN I-REG. (ODD), GO → PI → WI_2 AFTER PA OF ODD

ET TRANSFER IN I-REGISTER FLAG THE END OF XFER INSTR. (PA N OR PA ODD)

NOTE 1. TRANSFER IN I-REGISTER INCLUDES: RET, RTD, RCU, XEC, XED INSTRUCTION IN ADDITION TO ALL TRANSFERS.

NOTE 2. "GENERAL" INSTRUCTION FETCH DOES NOT INVOLVE INSTRUCTION FETCHES THAT ARE A RESULT OF SOME CONTROL FRAME AND SPECIAL CLASS INSTRUCTIONS.

NOTE 3. "GENERAL" INSTRUCTION FETCH SEQUENCE WILL BE ENTERED IF THE PREVIOUS-ODD INSTR. WAS NOT A CONTROL FRAME CLASS (I.E. TRANSFER, XEC, XED) OR SPECIAL CLASS INSTR. (I.E. RETURN, RTD, TSB_n.)

NOTE 4. REFER TO MCF GENERAL OPERATING CYCLE FIG. 9.2.5

FIG. 9.2.4 GENERAL INSTRUCTION FETCH SEQUENCE, FLOW CHART (sheet 2 of 2)

9.2.4 - contd

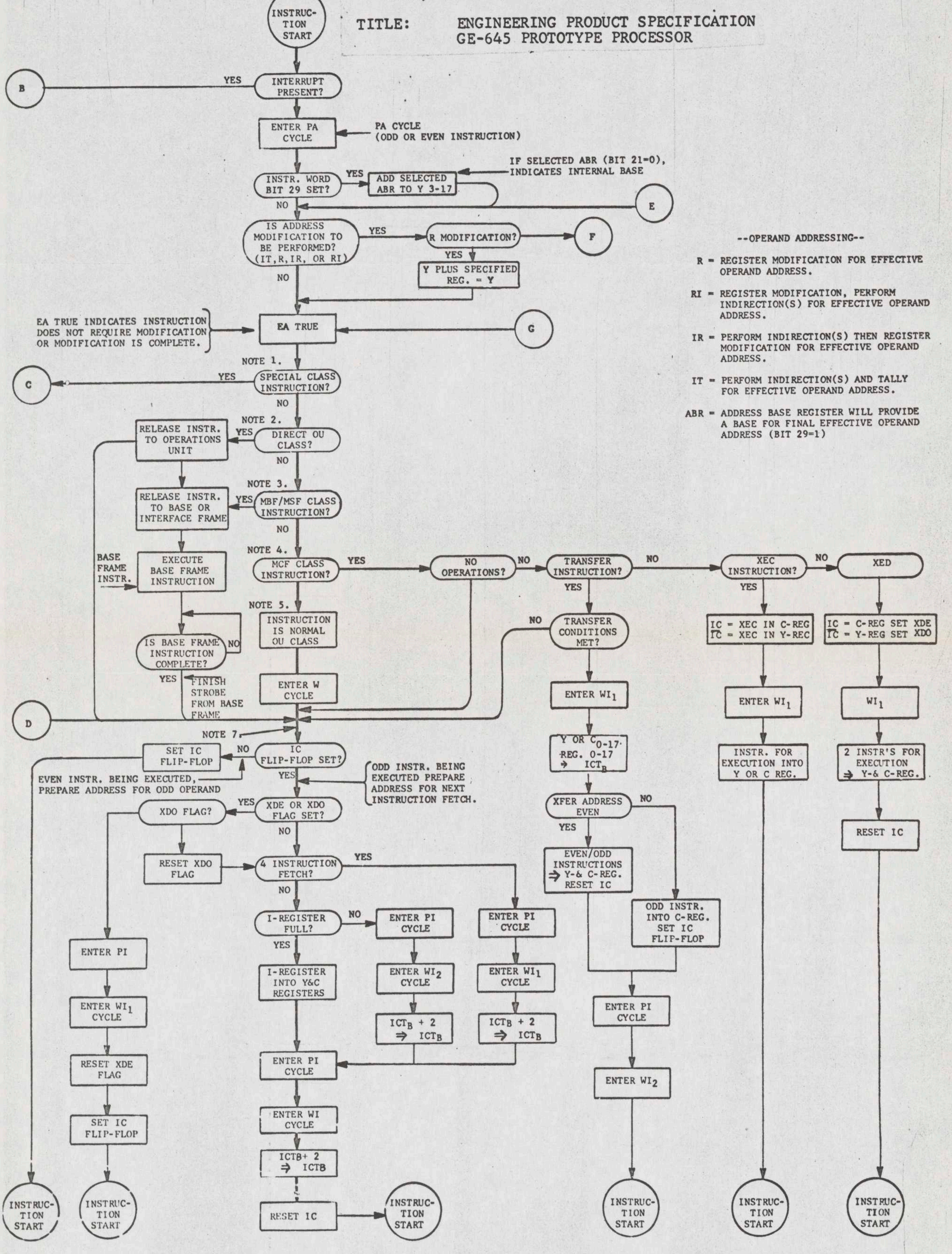
If the Y-, C- and I-registers are full, and there is a load-store instruction combination in the I-register, the fetching of a pair of instructions to fill the I-register after the I-register contents are moved into the working registers, is postponed until the time between the PA cycle for the load type instruction and the PA cycle for the store type instruction. This allows the Control Frame to take advantage of the relatively long time between the load- and store-instruction execution. When an XDE or XDO flag is set, indicating an execute double instruction is in the even or odd instruction register, the instruction fetch sequence is a little different. For an XED instruction in the even instruction location, two words are pulled from the address specified by the XED instruction during the PA cycle of the XED instruction, executed, then the instruction in the C-register that was destroyed by bringing in the second XEDed instruction is refetched. When an execute double instruction is in the odd instruction location, the XEDed instructions are fetched during the PA cycle of the XED instruction, executed, then four instructions must be pulled in the normal manner for a four-instruction fetch.

9.2.5 OPERAND FETCHES

As mentioned previously, operand fetches within the Control Frame fall into five general classes: Normal OU; Direct OU; Control Frame; Base Frame; and Specials. Instruction in each class all, except the last class Specials, cause the Control Frame to respond in essentially the same manner. Each of the instructions in the "Specials" class cause the Control Frame to react differently, depending on the instruction involved. Figure 9.2.5 is a very generalized flow chart of the major sequences involved for each of the five classes. Included in the table on Fig. 9.2.5 are the different instructions in each class. Generally speaking, each class involves going through the same sequences for each of the instructions of that class. The sequences can become quite involved and overlapped, since instruction and operand address preparation, in addition to operand

9.2.5 - contd

execution, can go on at the same time for a maximum of three instructions before the Operations Unit completes the first instruction. In addition, the possibility of faults occurring in various phases of the sequence have been left out. All the detailed sequences will be covered beginning in paragraph 9.4, in addition to how each "piece" of the sequence involves the hardware elements explained in paragraph 9.3.


INSTRUCTION START
INTERRUPT PRESENT?
YES
B
ENTER PA CYCLE
PA CYCLE (ODD OR EVEN INSTRUCTION)
INSTR. WORD BIT 29 SET?
YES
ADD SELECTED ABR TO Y 3-17
IF SELECTED ABR (BIT 21=0), INDICATES INTERNAL BASE
NO
E
IS ADDRESS MODIFICATION TO BE PERFORMED? (IT,R,IR, OR RI)
YES
R MODIFICATION?
F
YES
Y PLUS SPECIFIED REG. = Y
NO
EA TRUE INDICATES INSTRUCTION DOES NOT REQUIRE MODIFICATION OR MODIFICATION IS COMPLETE.
EA TRUE
G
NOTE 1.
SPECIAL CLASS INSTRUCTION?
YES
C
NO
NOTE 2.
DIRECT OU CLASS?
YES
RELEASE INSTR. TO OPERATIONS UNIT
NO
NOTE 3.
MBF/MSF CLASS INSTRUCTION?
YES
RELEASE INSTR. TO BASE OR INTERFACE FRAME
NO
NOTE 4.
MCF CLASS INSTRUCTION?
YES
NO OPERATIONS?
NO
TRANSFER INSTRUCTION?
NO
XEC INSTRUCTION?
NO
XED
BASE FRAME INSTR.
EXECUTE BASE FRAME INSTRUCTION
IS BASE FRAME INSTRUCTION COMPLETE?
NO
YES
FINISH STROBE FROM BASE FRAME
NOTE 5.
INSTRUCTION IS NORMAL OU CLASS
ENTER W CYCLE
D
NOTE 7.
YES
NO
TRANSFER CONDITIONS MET?
YES
ENTER WI1
Y OR C0-17 REG. 0-17 ⇒ ICTB
XFER ADDRESS EVEN
YES
NO
EVEN/ODD INSTRUCTIONS ⇒ Y-& C-REG. RESET IC
ODD INSTR. INTO C-REG. SET IC FLIP-FLOP
ENTER PI CYCLE
ENTER WI2
INSTRUCTION START
YES
IC = XEC IN C-REG
IC̅ = XEC IN Y-REC
ENTER WI1
INSTR. FOR EXECUTION INTO Y OR C REG.
INSTRUCTION START
IC = C-REG SET XDE
IC̅ = Y-REG SET XDO
WI1
2 INSTR'S FOR EXECUTION ⇒ Y-& C-REG.
RESET IC
INSTRUCTION START
SET IC FLIP-FLOP
NO
IC FLIP-FLOP SET?
EVEN INSTR. BEING EXECUTED, PREPARE ADDRESS FOR ODD OPERAND
YES
ODD INSTR. BEING EXECUTED PREPARE ADDRESS FOR NEXT INSTRUCTION FETCH.
XDO FLAG?
YES
XDE OR XDO FLAG SET?
NO
RESET XDO FLAG
4 INSTRUCTION FETCH?
YES
NO
I-REGISTER FULL?
NO
ENTER PI CYCLE
ENTER PI CYCLE
YES
I-REGISTER INTO Y&C REGISTERS
ENTER WI2 CYCLE
ENTER WI1 CYCLE
ICTB + 2 ⇒ ICTB
ICTB + 2 ⇒ ICTB
ENTER PI
ENTER WI1 CYCLE
RESET XDE FLAG
SET IC FLIP-FLOP
ENTER PI CYCLE
ENTER WI CYCLE
ICTB+ 2 ⇒ ICTB
RESET IC
INSTRUCTION START
INSTRUCTION START
INSTRUCTION START

--OPERAND ADDRESSING--

R = REGISTER MODIFICATION FOR EFFECTIVE OPERAND ADDRESS.

RI = REGISTER MODIFICATION, PERFORM INDIRECTION(S) FOR EFFECTIVE OPERAND ADDRESS.

IR = PERFORM INDIRECTION(S) THEN REGISTER MODIFICATION FOR EFFECTIVE OPERAND ADDRESS.

IT = PERFORM INDIRECTION(S) AND TALLY FOR EFFECTIVE OPERAND ADDRESS.

ABR = ADDRESS BASE REGISTER WILL PROVIDE A BASE FOR FINAL EFFECTIVE OPERAND ADDRESS (BIT 29=1)

FIG. 9.2.5 Operand Fetch, Flowchart

(Page 1 of 3)

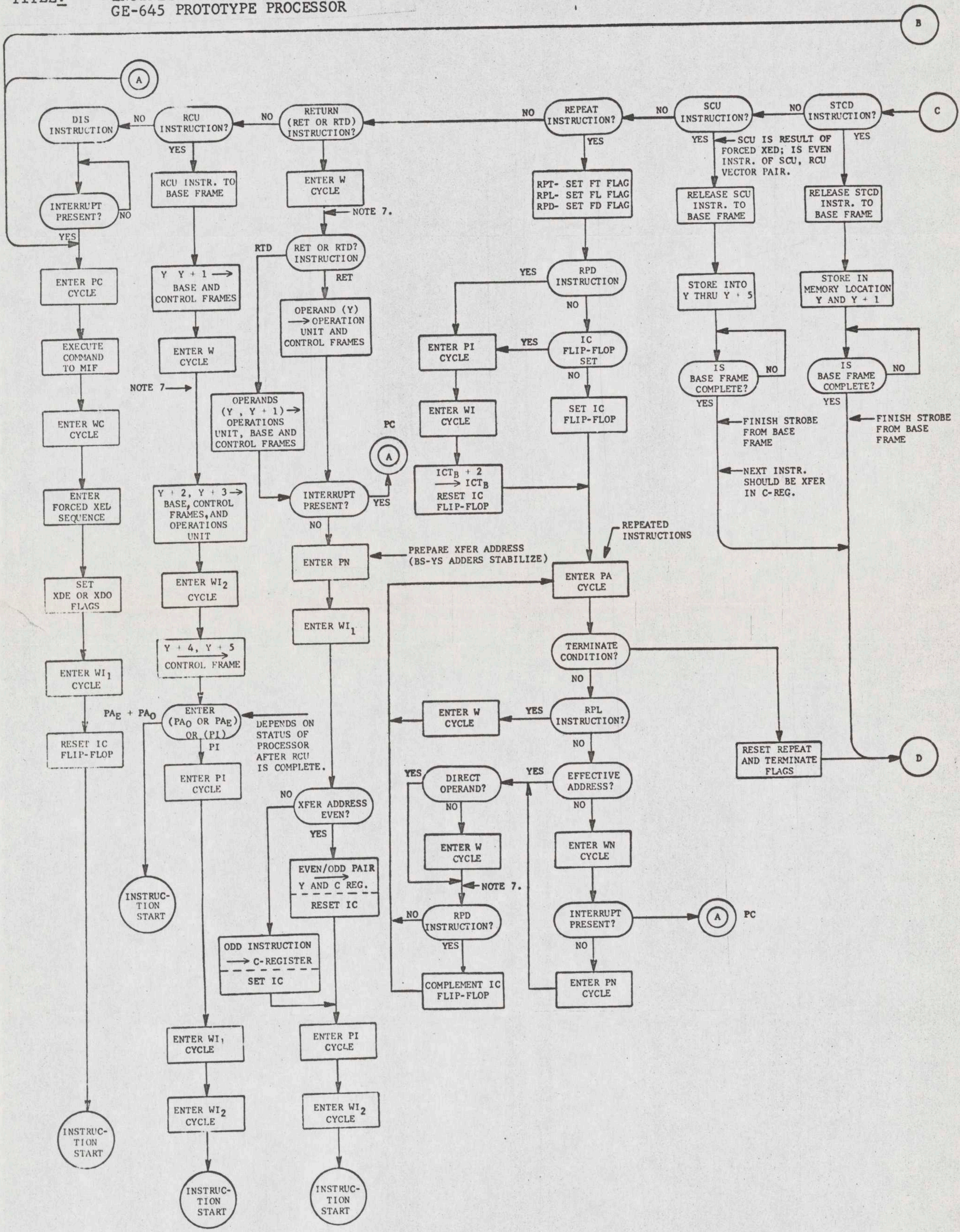
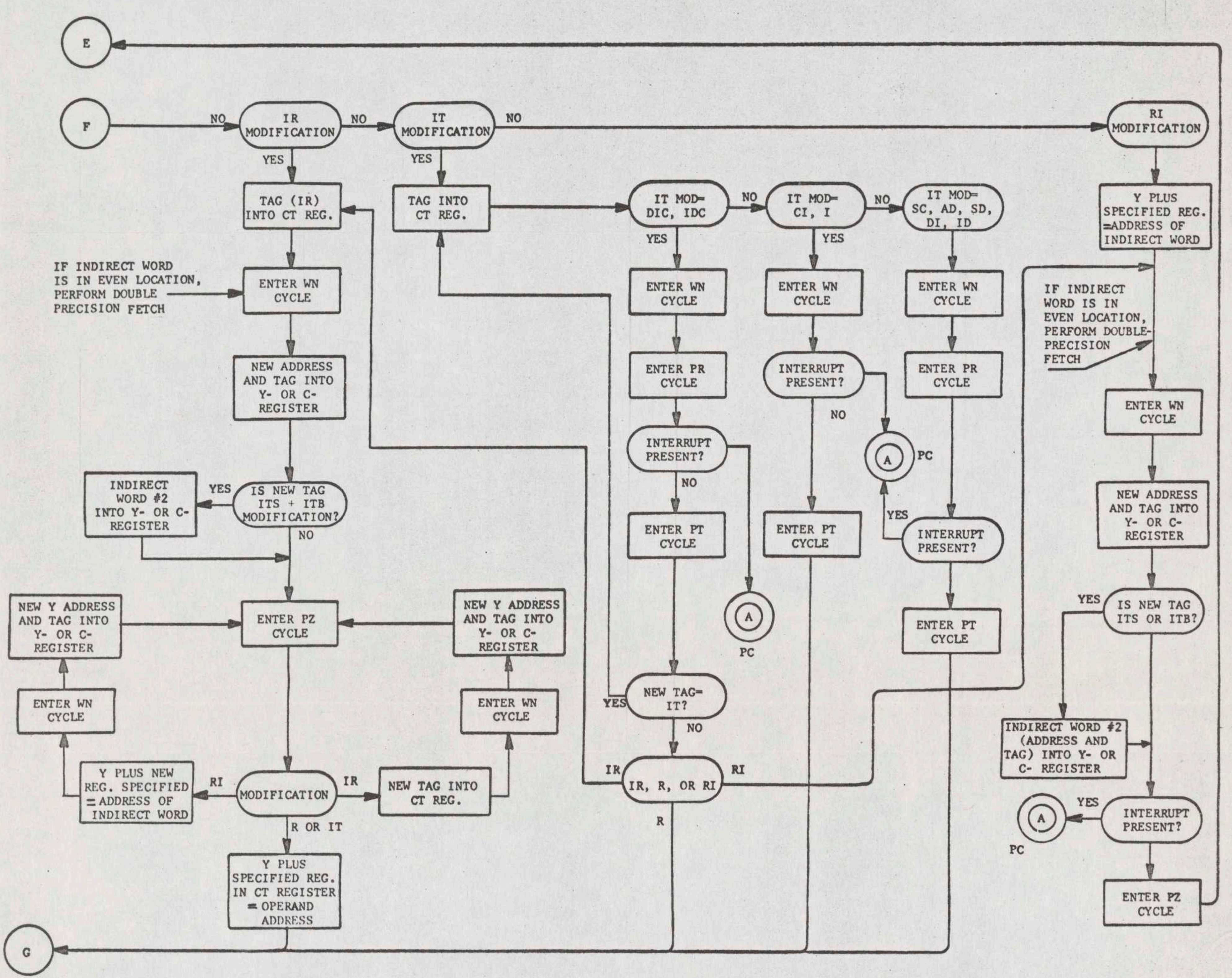

B
A
C
STCD INSTRUCTION?
NO
YES
SCU INSTRUCTION?
NO
YES
SCU IS RESULT OF FORCED XED; IS EVEN INSTR. OF SCU, RCU VECTOR PAIR.
REPEAT INSTRUCTION?
NO
YES
RETURN (RET OR RTD) INSTRUCTION?
NO
RCU INSTRUCTION?
NO
YES
DIS INSTRUCTION
INTERRUPT PRESENT?
NO
YES
ENTER PC CYCLE
EXECUTE COMMAND TO MIF
ENTER WC CYCLE
ENTER FORCED XEL SEQUENCE
SET XDE OR XDO FLAGS
ENTER WI1 CYCLE
RESET IC FLIP-FLOP
INSTRUCTION START
RCU INSTR. TO BASE FRAME
Y Y + 1 → BASE AND CONTROL FRAMES
ENTER W CYCLE
NOTE 7
Y + 2, Y + 3 → BASE, CONTROL FRAMES, AND OPERATIONS UNIT
ENTER WI2 CYCLE
Y + 4, Y + 5 → CONTROL FRAME
ENTER (PAO OR PAE) OR (PI)
PAE + PAO
PI
DEPENDS ON STATUS OF PROCESSOR AFTER RCU IS COMPLETE.
INSTRUCTION START
ENTER PI CYCLE
ENTER WI1 CYCLE
ENTER WI2 CYCLE
INSTRUCTION START
ENTER W CYCLE
NOTE 7.
RET OR RTD? INSTRUCTION
RTD
RET
OPERAND (Y) → OPERATION UNIT AND CONTROL FRAMES
OPERANDS (Y , Y + 1) → OPERATIONS UNIT, BASE AND CONTROL FRAMES
INTERRUPT PRESENT?
YES
PC
A
NO
ENTER PN
PREPARE XFER ADDRESS (BS-YS ADDERS STABILIZE)
ENTER WI1
XFER ADDRESS EVEN?
NO
YES
EVEN/ODD PAIR Y AND C REG.
RESET IC
ODD INSTRUCTION → C-REGISTER
SET IC
ENTER PI CYCLE
ENTER WI2 CYCLE
INSTRUCTION START
RPT- SET FT FLAG
RPL- SET FL FLAG
RPD- SET FD FLAG
RPD INSTRUCTION
YES
NO
IC FLIP-FLOP SET
YES
NO
ENTER PI CYCLE
ENTER WI CYCLE
ICTB + 2 → ICTB
RESET IC FLIP-FLOP
SET IC FLIP-FLOP
REPEATED INSTRUCTIONS
ENTER PA CYCLE
TERMINATE CONDITION?
NO
RPL INSTRUCTION?
YES
ENTER W CYCLE
NO
EFFECTIVE ADDRESS?
YES
NO
DIRECT OPERAND?
YES
NO
ENTER W CYCLE
NOTE 7.
RPD INSTRUCTION?
NO
YES
COMPLEMENT IC FLIP-FLOP
ENTER WN CYCLE
INTERRUPT PRESENT?
NO
A
PC
ENTER PN CYCLE
RESET REPEAT AND TERMINATE FLAGS
D
RELEASE SCU INSTR. TO BASE FRAME
STORE INTO Y THRU Y + 5
IS BASE FRAME COMPLETE?
NO
YES
FINISH STROBE FROM BASE FRAME
NEXT INSTR. SHOULD BE XFER IN C-REG.
RELEASE STCD INSTR. TO BASE FRAME
STORE IN MEMORY LOCATION Y AND Y + 1
IS BASE FRAME COMPLETE?
NO
YES
FINISH STROBE FROM BASE FRAME

FIG. 9.2.5 Operand Fetch, Flowchart

(Page 2 of 3)


E
F
NO
IR MODIFICATION
NO
IT MODIFICATION
NO
RI MODIFICATION
YES
YES
TAG (IR) INTO CT REG.
TAG INTO CT REG.
IT MOD= DIC, IDC
NO
IT MOD= CI, I
NO
IT MOD= SC, AD, SD, DI, ID
Y PLUS SPECIFIED REG. =ADDRESS OF INDIRECT WORD
YES
YES
IF INDIRECT WORD IS IN EVEN LOCATION, PERFORM DOUBLE PRECISION FETCH
ENTER WN CYCLE
ENTER WN CYCLE
ENTER WN CYCLE
ENTER WN CYCLE
IF INDIRECT WORD IS IN EVEN LOCATION, PERFORM DOUBLE-PRECISION FETCH
NEW ADDRESS AND TAG INTO Y- OR C- REGISTER
ENTER PR CYCLE
INTERRUPT PRESENT?
ENTER PR CYCLE
ENTER WN CYCLE
NO
INTERRUPT PRESENT?
A
PC
NEW ADDRESS AND TAG INTO Y- OR C- REGISTER
INDIRECT WORD #2 INTO Y- OR C- REGISTER
YES
IS NEW TAG ITS + ITB MODIFICATION?
NO
NO
YES
ENTER PT CYCLE
ENTER PT CYCLE
INTERRUPT PRESENT?
NEW Y ADDRESS AND TAG INTO Y- OR C- REGISTER
ENTER PZ CYCLE
NEW Y ADDRESS AND TAG INTO Y- OR C- REGISTER
A
PC
YES
IS NEW TAG ITS OR ITB?
ENTER PT CYCLE
ENTER WN CYCLE
ENTER WN CYCLE
YES
NEW TAG= IT?
NO
INDIRECT WORD #2 (ADDRESS AND TAG) INTO Y- OR C- REGISTER
Y PLUS NEW REG. SPECIFIED = ADDRESS OF INDIRECT WORD
RI
MODIFICATION
IR
NEW TAG INTO CT REG.
IR
IR, R, OR RI
RI
A
PC
YES
INTERRUPT PRESENT?
R OR IT
R
Y PLUS SPECIFIED REG. IN CT REGISTER = OPERAND ADDRESS
ENTER PZ CYCLE
G

--NOTES--

1. SPECIAL CLASS INSTRUCTIONS ARE THOSE WHICH INVOLVE MULTIPLE FRAMES AND INDIVIDUALLY DETERMINE THE MCF SEQUENCE MODE.

2. DIRECT OU CLASS INSTRUCTIONS ARE THE OU INSTRUCTIONS THAT DO NOT REQUIRE MEMORY ACCESS FOR OPERAND.

3. MBF/MSF CLASS INSTRUCTIONS ARE THOSE WHICH INVOLVE THE BASE AND/OR ASSOC. MEMORY FRAMES, AND FOR THE MOST PART ARE EXECUTED OUTSIDE THE CONTROL FRAME.

4. MCF CLASS INSTRUCTIONS ARE THOSE WHICH ARE EXECUTED PRIMARILY WITHIN THE CONTROL FRAME.

5. NORMAL OU CLASS ARE THOSE OU INSTRUCTIONS WHICH REQUIRE AN OPERAND TO BE FETCHED FROM OR SENT TO MEMORY.

SPECIAL	DIRECT OU	MBF/MSF		MCF	NORMAL OU
SCU *	SHIFTS	ADB$_n$		TRANSFERS	ARITHMETIC
STCD *	GTB	EAB$_n$		NOP	OU LOAD'S
RCU *	NEG	EAP$_n$		XEC	OU STORE'S
RET	NEGL	LBR$_n$		XED	LOGICAL
RTD *	FNO	LDB	MBF	TSB$_n$ *	BCD
	FNEG	LDBR			
CIOC	EAA	LDCF			RMCM
MME 1-4	EAQ	SBR$_n$			SMCM
DRL	EAX$_n$	SDBR			SZN
RPT		STB			LXL$_n$
RPD		STP$_n$			SXL$_n$
DIS		CAM			
		SAZ	MSF		
		SAM			
* INVOLVE BASE FRAME					

6. HEAVY LINE INDICATES "NORMAL" PATH OF CONTROL FRAME OPERATION, FROM INSTRUCTION INITIATION TO SUBSEQUENT INSTRUCTION FETCH. "NORMAL" OPERATING CYCLE IS ASSUMING A NORMAL OU INSTR. WITH NO ADDRESS MODIFICATION.

7. OPERAND IS FETCHED OR STORED FOR OU DURING W CYCLE. CONTROL FRAME IS AVAILABLE FOR NEXT INSTRUCTION/OPERAND FETCH AT END OF W CYCLE.

FIG. 9.2.5 Operand Fetch, Flowchart

(Page 3 of 3)

9.3 PRINCIPLE HARDWARE ELEMENTS AND FUNCTIONS

The following paragraphs explain the operation of the principle hardware elements and functions in the Control Frame that can be defined by a definite hardware physical relationship (for example, a register, counter, etc.) or a functional grouping of circuits (for example, strobe generation circuits and interface term logic, Z-function, etc.). The sequence of operation and relationship between all elements and functions is covered in detail in paragraph 9.4.

9.3.1 STROBE \$R, \$C, \$ΔAM1, \$ΔAM2, \$IF, AND INTERFACE TERM (See 43D163387, sheets 2,3,4)

Strobes \$C and \$R are the two most widely used strobes in the Control Frame. Strobe \$C is utilized more extensively than any other strobe and is generally concerned with the setting and resetting of the sequence flags (PA, PZ, PN, PT, IC etc.); while strobe \$R occurs only if an interrupt is sent to the Memory Controller. Strobe \$R is used to set and reset the flags concerned with the interrupt process. Strobes \$ΔAM1, \$ΔAM2, and \$IF are used to provide time delays to: enable instruction decodes and the Z (Store-Compare) function to stabilize; allow the BS-adder to stabilize; start the Interface Frame; and generally hold up the development of further strobes until certain functions in other frames have had a chance to be performed and become stabilized. The functions of strobes \$ΔAM1, \$ΔAM2, and \$IF will become clearer during the discussion of the interface term (a composite equation), which is developed from a number of conditions indicating whether all of the affected areas within the Processor are ready to proceed with certain sequences. Since the timing in many areas of the Processor is asynchronous, the interface equation is quite large. For this reason the interface equation is explained in detail in paragraph 9.3.1.2.

9.3.1.1 Strobe $C and $R Generation (See 43D163387, sheet 2)

An examination of the two pulse drivers that generate $C and $R show that they have three common signal inputs, $ABT, $PRI, and $PRII. In addition, $C may be generated by $CDA or $DS·PR). Thus it should be clear that $C may occur alone, but $R may never occur without $C also being generated.

Strobe C - Strobe $C may be generated under any one of the five following conditions:

1) Receipt of $CDA, which is a pulse derived from the data-in control logic used to load the Y- and C-registers (see 43D163387 sheet 24). For a normal instruction fetch, when the Y-register is loaded with the even instruction of a pair:

$$\$CDA = \$DA_1 \cdot DYC_{\Delta} \cdot (\text{PR or no P cycle})$$

Level DYC_{Δ} is from a flip-flop which sets whenever an instruction word is to be brought into the Y- or C-registers. Strobe $DA, is generated in the Control Frame after its reception from the Interface Frame, and indicates that data is on the ZDI lines from memory.

2) Strobe $DS and PR are present. These terms will be true whenever the Control Frame is in the PR cycle for some IT address modification. Level PR is the prepare rewrite flag, and $DS is from the Interface Frame indicating that data has been loaded into the Data-Out Register in the Interface Frame, to be later stored in memory.

3) Strobe $ABT is received from the base frame fault logic as a result of the WF (Wait for Fault) flip-flop in the Base Frame being set. The WF level indicates that either an external interrupt has occurred and will be recognized, or a fault has occurred. If WF is set, the Control Frame will have already performed a "snapshot" routine to safe-store the current status of the Control Unit. The "snapshot" sequence is explained in detail in later paragraphs. With the receipt of $ABT, $C and $R are generated to allow an interrupt to be set to the Memory Controller via the Interface Frame to complete the interrupt routine.

9.3.1.1 - contd

4&5) Strobes $PRI or $PRII are received. These strobes are nearly identical in their source. However, $PRI is inhibited if the Interface Frame has detected a fault; whereas $PRII is generated only if a fault occurs.

$$\$PRI\ (\text{See 43D163380 sheet 6}) = \$PRC \cdot \overline{APE} \cdot \overline{ISR} \cdot \overline{SPF} + \$TEST \cdot RST$$

where:

$$\$PRC = IFR \cdot SPR \cdot BFR \cdot \overline{RR} \cdot CFR$$

and (See 43D163387 sheet 2)

$$\$PRII = IFR \cdot SPR \cdot BFR \cdot RR \cdot CFR$$

The similarity at the orignation of $PRC and $PRII is readily apparent since $\overline{RR}$ and RR are the only differences. The term RR will be true for certain faults detected by the base frame fault logic and will inhibit $PRC, but will allow $PRII. This is to allow the Control Unit to complete certain procedures even though the rest of the machine will be halted. Strobe $PRII will generate a $R to enable the Operations Unit to complete its operations, but $INT to the Memory is inhibited. If $PRC is allowed to go to the Interface Frame ($\overline{RR}$ present), then the Interface Frame can still inhibit $R and also $INT for a fault. The terms that must be present to generate $PRI from $PRC in the Interface Frame include $\overline{APE}$ (not appending parity error), $\overline{ISR}$ (not inhibit $R), and $\overline{SPF}$ (part of the manual program step circuits for test purposes). The terms for $TEST and RST enable the generation of $PRI in the test mode of operation where the Processor is being manually stopped. When $PRC is not inhibited in the Interface Frame, it emerges a $PRI, which subsequently generates $C and $R resulting in an interrupt to continue the normal sequence.

9.3.1.1 - contd

Re-examining the gated pulse amplifiers which generate $PRC and $PRII, one finds that they are identical except for $\overline{RR}$ and RR, which has been explained. The other terms concerned are:

IFR -- interface frame ready, which is set by $IFF from the Interface Frame and generally is the last condition to be brought up. Strobe $IFF specifies that the Interface Frame has a final address and is ready to send a $INT to memory.

SPR -- this level is a flip-flop used only for test and step purposes and will normally be set.

BFR -- base frame ready flip-flop, which is set by $BC + $PRR ($\overline{PN}$ + $\overline{BFI}$) + ($\overline{EA}$·BFI·$PRR). Strobe $BC comes from the Base Frame and signals the completion of a base frame instruction. Strobe $PRR is generated by the interface term coming true. If there is not a base frame instruction, the BFR is set almost as soon as the Control Frame starts a preparation cycle. Finally, if there is a base frame instruction but the effective address has not yet been prepared, $PRR will set BFR. The BFR flip-flop may not be set by $BC if any indirections or appending is required, since the Base Frame is not started by the Interface Frame until an EA·FA are present and hence no $BC to set BFR is generated.

CFR -- control frame ready is the last term necessary and is a direct result of the interface term (INTR) coming true. CFR is almost always the first flip-flop set, except if $\overline{BFI}$ is present. Strobe $PR unconditionally sets CFR.

$PR = $PCFF (54 or 126 nanoseconds after $PCFF)·$\overline{CSR}$

where:

$PCFF = $PRR (36 or 18 nanoseconds after $PRR)·$\overline{ONC}$

and

$PRR = INTR (Interface term via a 301 pulse amplifier)

9.3.1.1 - contd

The times in the parentheses for $PR are the differences between the time that a $LAST from the Operations Unit is being processed by the Control Unit, and no $LAST is received; the longer time being for $LAST. Strobe $LAST is used to generate a strobe to count the TMCH counter down. The TMCH counter is used to keep track of the number of operation codes sent to the Operations Unit for execution. The possibility exists that the TMCH counter can be counted up and down at the same time. Hence, the longer delay for $PR, when SLO/FST is set, to provide time for possible counting of TMCH down before being counted up. The conditions involved are covered in more detail in the TMCH counter discussion paragraph 9.3.9. The term $\overline{CSR}$ is normally true unless a fault is detected or a fault operation code is detected after the interface equation has been satisfied.

The times shown in parenthesis for $PCFF are the difference between the Control Unit in fast clock (determined by Maintenance Panel switch settings) and normal clock operation. The shorter time is used only for marginal timing tests of the Control Unit. If an Operation Not Complete fault is detected $\overline{ONC}$ is false and $PCFF is inhibited.

Strobe $R -- As mentioned previously, $R is generated from $ABT, $PRI or $PRII. These three conditions are the only ones that will generate $R, and their meanings have already been explained under $C generation.

9.3.1.2 Interface Term (INTR) (See 43D165387, sheet 4)

The interface term combined equation (INTR) will become true if the required portions of the Processor are ready to proceed. This term may be brought up by any one of nine different enable conditions. The composite equation will be separated here for clarity, and the nine possible enable conditions will be examined first in the manner in which they produce INTR true [illegible] (provided INTR is not inhibited, see 9.3.1.3):

9.3.1.2 Interface Term (INTR) (See 43D163387, sheet 4)

The interface term combined equation (INTR) will become true if the required portions of the Processor are ready to proceed. This term may be brought up by any one of nine different enable conditions. The composite equation will be separated here for clarity, and the nine possible enable conditions will be examined first in the manner in which they produce INTR true (BINTIB RNSO) (provided INTR is not inhibited, see 9.3.1.3):

1) $(\mathrm{TMCH} = 0) \cdot (\overline{\mathrm{EA}) \cdot (\mathrm{STORES} + \mathrm{RAR}})$

If the TMCH counter equals zero the Operations Unit has completed its last instruction and the current instruction is not an Operations Unit store or read-alter-rewrite. (In other words, any instruction except stores or read-alter-rewrites.)

2) $\mathrm{EA} \cdot (\mathrm{STORES} + \mathrm{RAR}) \cdot \mathrm{SDY}$

An effective address is present for a store or read-alter-rewrite operation. The SDY term indicates that the Operations Unit has the data for the store operation, and the Control Frame may now begin to prepare for having the Interface Frame issue an eventual $INT to Memory.

3) $(\overline{\mathrm{XDE} + \mathrm{XDO}}) \cdot \mathrm{PI}$

The Control Frame is in a normal PI cycle, and the execute double even, and execute double odd flip-flops are reset indicating that the PI cycle is not the result of an XED instruction.

4) $\mathrm{TERM} \cdot \overline{\mathrm{CSR}}$

TERM is a level brought high when the Operations Unit terminates a "repeat" instruction. It may also be true if a fault condition occurs and the Processor does not request a memory cycle. The term CSR is normally not present unless a faulty operation code or a fault is detected and the processor fault logic has not had a chance to initiate a fault sequence. When CSR sets, $PR is stopped and the

9.3.1.2 - contd

interface term is held off. For "repeat" instructions, when TERM is brought up during a fault initiated by SFF, SFF is forced off by TERM as long as the fault is not an Operations Unit fault. In other words, unless the fault is an Operations Unit fault and would effect the execution of the "repeated" instruction(s) it is ignored. As soon as CSR is set and pulls down SFF in conjunction with a TERM condition, CSR is forced reset allowing the Interface Term and $PR (other conditions being equal).

5) $(EA)\cdot(Y+RSW)\cdot\overline{TAS}\cdot\overline{RPT'S}\cdot\overline{MFREE1}\cdot DBF\cdot(TMCH=1)\cdot\overline{BFI}$ EA·(Y+RSW) -- indicates that an effective address for a direct operand is present; Y signifies the direct operand and, RSW is the read switches instruction which is also a direct operation.

$\overline{TAS}$ - indicates a "not test and step" condition.

$\overline{RPT'S}$ - means that the Control Frame cannot be doing any of the "repeat" instructions.

$\overline{MFREE1}$ - will be reset if the Operations Unit is ready to accept an operand.

$\overline{DBF}$ - indicates that a direct operand has not been loaded into the Direct Buffer in the Base Frame. The Direct Buffer holds the direct operand.

(TMCH = 1) - Indicates that the Operations Unit is still operating on the previous instruction but allows a direct operand to be loaded in the Direct Buffer even though the Operations Unit may be busy. If the Direct Buffer had been full DBF would be true and stop further action until the Operations Unit takes the operand.

$\overline{BFI}$ - This term is to prevent any overlap of Operations Unit and base frame instructions. If the Operations Unit is busy, the interface term is held off until TMCH = 0; then released (equation 1) to allow the base frame instruction to be completed.

9.3.1.2 - contd

6) $(\mathrm{PA}+\mathrm{PZ})\cdot(\overline{\mathrm{ZI}=\mathrm{IT}})\cdot(\overline{\mathrm{FT}+\mathrm{FD}})\cdot(\overline{\mathrm{EA}})\cdot(\overline{\mathrm{CT}+\mathrm{DIS}})$

The Processor is preparing an indirect address (PA) or (PZ) of the IR or RI type. Terms $\overline{\mathrm{FT}+\mathrm{FD}}$ indicate not a repeat or repeat double instruction; EA has not been generated and there is not a conditional transfer or delay until interrupt instruction on the operation decode bus $(\overline{\mathrm{CT}+\mathrm{DIS}})$.

7) $(\mathrm{SFF})\cdot(\mathrm{LPC})\cdot(\mathrm{DVK}+\mathrm{FLF}+\mathrm{SEP}+\mathrm{BTPFOB})$

This group of terms is for conditions where the Control Unit is waiting to service an Operations Unit or Control Unit fault. The terms DVK, FLF and SEP, are for Divide Check, Overflow and Parity, Operations Unit faults. Control unit instructions that cause faults are handled by BTPFOB which contains TMCH = 0. The idea behind equation 7, is that for Operations Unit and Control Unit faults, the Control Unit is allowed to continue and service the fault.

8) $(\mathrm{TMCH}=1)\cdot(\mathrm{EA}\cdot\mathrm{NOP}\cdot\overline{\mathrm{ONR}})\cdot(\mathrm{ZI}=\overline{\mathrm{IT}})$

These terms are to allow the Control Unit to do a "no operation" (NOP) instruction, $\mathrm{ZI}=\overline{\mathrm{IT}}$ is to inhibit the interface term for IT modifier and $\overline{\mathrm{ONR}}$ inhibits the interface term until the 0-register in the Operations Unit is free.

9)
$$
\begin{aligned}
&[(\mathrm{TMCH}=1)\cdot(\overline{\mathrm{FT}\cdot\mathrm{RF}+\mathrm{FL}}) && (9a)\\
&+(\overline{\mathrm{LREG}}\cdot(\overline{\mathrm{FT}+\mathrm{FD}+\mathrm{FL}})\cdot\overline{\mathrm{TMCH}=2}\cdot(\overline{\mathrm{XDE}}+\mathrm{XDO})\cdot \mathrm{Z2}_4=\mathrm{SHORT}] && (9b)\\
&\cdot[(\overline{\mathrm{TAS}}\cdot\mathrm{T}\cdot\overline{\mathrm{STORE}+\mathrm{RAR}\cdot\mathrm{RET}}\cdot\overline{\mathrm{EA}\cdot\mathrm{Y}+\mathrm{RSW}}) && (9c)\\
&\cdot(\overline{\mathrm{BFI}}) && (9d)\\
&\cdot(\overline{\mathrm{MDV}\cdot(\mathrm{ZI}}=\overline{\mathrm{IT}})\cdot\mathrm{EA} && (9e)\\
&\cdot(\overline{\mathrm{DBF}\cdot\mathrm{MFI}})] && (9f)
\end{aligned}
$$

This group of terms allows the interface to become true and produce an eventual \$R, and subsequent interrupt for an Operations Unit operand, in those cases where the Operations Unit already has one or two operands. The assumption is to allow another operand to be sent from memory to the Operations

9.3.1.2 - contd

Unit, because the Operations Unit will be finished with it's current operation by the time the operand arrives.

9a) This equation represents the first case: where the Operations Unit has one operand and is not in the process of executing a Repeat Link operation, or the first equations (9c) through (9f) are essentially the inhibit PA cycle of a repeated instruction conditions for the first case.

9b) The second case is where the Operations Unit has two operands but a third operand can be sent under certain conditions. Equations (9c) through (9f) still represent the inhibit conditions. The key to this equation is TMCH=2 which indicates that the Operations Unit has two instructions currently being executed. Normally it could be assumed that no other instruction would be allowed until TMCH=0 or 1. However, $Z2_4$=SHORT is a register which sets when the first of two instructions sent to the Operations Unit is about to be completed. SHORT indicates that is is a type instruction. such as a LDA instruction which requires only a GS, GF cycle in the Operations Unit (see Operations Unit write-up for definitions of the GS and GF cycles.) The assumption is that even though TMCH=2, the interface term may be brought up, because by the time the Control Frame is ready, the short operations unit instruction will have been completed. Other restrictions on this situation are: the instructions currently being processed by the Control Unit may not be Load Registers, any "repeat," or an execute double instruction in the even or odd instruction register (Y or C). If the instruction being processed was any of the above, then the Control Frame could be ready before the Operations Unit was finished.

9.3.1.2 - contd

9c) This portion of the equation indicates that the Operation Unit is not in test and step, that all operations are Operations Unit types, that the current instruction is not a store or RAR, or there is not a return instruction or direct operation.

9d) When a base frame instruction is being processed, the Control Unit may not proceed until the Operations Unit completes it's operation so $\overline{\mathrm{BFI}}$ inhibits INTR.

9e) No multiply or divide operation can take place and no IT modification is allowed. An IT modification may alter a memory location or effect prior conditions within the Operations Unit, so an IT modification is not completed until the Operations Unit is finished with all instructions.

9f) If the Direct Buffer in the Base Frame is full and the M-register is full in the Operations Unit, hold up the interface term since the Operations Unit will not be able to accept an operand.

9.3.1.3 Interface Term Inhibits

The previous nine equations define the conditions that cause the interface term to come true and allow the Control Unit to proceed. Seven conditions can inhibit the interface term. These conditions are defined by the state of five flip-flops and two logic levels.

INTR not true = 1 (Refer to 43D163387, sheet 4)

1)
$$
\begin{aligned}
\text{PIN flip-flop set} &= \$\mathrm{PIN}\cdot(\mathrm{BSIN1B}) \\
&\quad + \mathrm{STOP} \\
&\quad + \mathrm{ONC} \\
&\quad + (\mathrm{WF}\cdot\overline{\mathrm{PC}}\cdot\overline{\mathrm{MASF}}) \\
\text{reset} &= \$\mathrm{INT}\cdot(\mathrm{FA}+\mathrm{PC})
\end{aligned}
$$

This flip-flop sets with a \$PIN from the Memory Controller which signals the Processor that the Memory Controller has acknowledged the interrupt and specific command. The enable level (BSIN1B) is a combination of the DPIN flip-flop (for double-precision words and the DPIN) level from the

---

9.3.1.3 - contd

Interface Frame. The PIN flip-flop is set with the second $PIN received from the Memory Controller during double-precision operations. When the PIN flip-flop is set, the indirect word, operand, or external interrupt. The terms STOP, ONC and $WF \cdot \overline{PC} \cdot$ MASF are to force set the PIN flip-flop for initializing the Processor if an operation is not complete, or if a fault condition is waiting to be serviced.

2) D flip-flop set = $\$\Delta AM2 \cdot \overline{WC}$
reset = $\$ PR + WF$

The D flip-flop is used to inhibit the interface term until the ZI (operation code) bus, and YS- and BS-adders have stabilized. Strobe $\$\Delta AM2$ sets D about 400 nanoseconds after one of three possible conditions occur which generate it, (see 43D163387, sheet 3).

$\$\Delta AM2$ = [QIPN1B (a prestrobe $\$IPR$ for external interrupts) (2a)
+ $\$C \cdot$ ¢Set P-cycles (2b)
$\cdot [\overline{ITS + ITB}] \cdot [\overline{FAST} + FAST]$ (2c)

2a) Strobe QIPN1B is used to generate $\$\Delta AM2$ any time that the Processor is in a DIS state and a Group VI fault or an external interrupt is detected. Group VI faults are: Timer Runout, Connect and Shutdown. Strobe $\$\Delta AM2$ is used to set the D flip-flop on the interface term equation to eventually allow the generation of $\$C$ starting the Processor and subsequently servicing of the fault.

2b) Any prepare address cycle, except PR, in addition to $\$C$, will generate $\$\Delta AM2$.

2c) Whenever an ITS or ITB modifier is detected, then the Control Unit must stop address preparation until the second indirect word is brought into the Y or C-register. The term $\overline{FAST}$ is from the Maintenance Panel for normal CF clock or fast CF clock. FAST and $\overline{FAST}$ cause $\$\Delta AM2$ to be speeded up by 18 nanoseconds (FAST) or at normal speed ($\overline{FAST}$).

9.3.1.3 - contd

One additional strobe is generated, as a function of $ΔAM1 and $ΔAM2; strobe $IF. Strobe $IF is used to start the Interface Frame, and is part of the conditions for generating $C and $R, ($IF·(SAB+SAP)= IFR reset, which stops $PRC and $PRII for $C and $R). Strobe $IF can be generated in one of four ways as shown by the terms in the brackets in the following equation for $IF.

$$
\begin{aligned}
\$IF = [&QSPEIB\cdot(\overline{DZ})\cdot ITS + ITB \cdot BZFIOB && (2d)\\
&BZFIOB = (\overline{FD+FL+FT\cdot OU\ stores\cdot PC\cdot EA})\\
&\quad\cdot(\overline{ZI = RI\cdot TERM\cdot RR})\\
&\quad\cdot(\overline{PC})\\
&\quad\cdot\ \overline{CT + DIS}]\\
+ [&\overline{EIP}\cdot CT\cdot\overline{PC}\cdot TRGO_1\cdot TMCH=0) && (2e)\\
+ (&FD+FL+FT\cdot OU\ stores\cdot\overline{PC}\cdot EA)\cdot SDY] && (2f)\\
+ [&QSDZ1B\ (\overline{ITS+ITB}\cdot BZFIOB] && (2g)
\end{aligned}
$$

2d) This equation represents the general case for generating strobe $IF. All the barred terms represent conditions for which $IF is inhibited: no Z-function, ITS or ITB address modification, repeated Operations Unit stores, interrupts, conditional transfers, or Processor in a DIS state.

2e) This group of terms are for the generation of $IF during transfers. The Operations Unit must be finished and no interrupt present before allowing $IF. Level $\overline{EIP}$ is used to inhibit starting $IF until some of the decodes involved have settled.

2f) Strobe $IF is regenerated concurrent with SDY for completing a sequence of repeated Operations Unit stores, that had been interrupted.

2g) When the Z-function level disappears, indicating that Operations Unit registers are now available for operand address preparation designated by the instruction tag the DZ flip-flop is reset. Resetting is accomplished by $LAST from the Operations Unit generating a $DOWN which resets DZ. When DZ is reset a pulse is generated which regenerates $IF.

9.3.1.3 - contd

3) SPC flip-flop set = PC
reset = $\$C\Delta \cdot WC$
+ (TERM) + (WF·FAULT)
+ HLT

This flag when set indicates that the Processor is preparing an execute interrupt cycle to service an external interrupt. SPC is reset with a $\$C^{\Delta} \cdot WC$ generated as a result of the execute interrupt information coming from memory during a wait for execute cycle. Resetting of SPC is also accomplished whenever the Processor is servicing a fault or terminate condition.

4) DZ flip-flop set = $Z \cdot \$\Delta AM_1$
reset = $DOWNΔ

This flag when set indicates that the Processor is waiting for a Z-function condition to be cleared. As soon as it is cleared INTR is enabled.

5) WI_1 flip-flop

This flag indicates that an instruction pair will be loaded into the Y- and C-registers. As soon as they are loaded, WI_1 is reset and INTR is enabled.

6) (CU/OU OLAPOFF + TERM + BFLAOB)·TST·$\overline{TMCH = 0}$

The terms are for control of the interface term from the Maintenance Panel. When the CU/OU OVERLAP OFF switch is on, and the Processor is in a test and step mode, then the interface term is held off until TMCH = 0. For cases where the TMCH counter will not go to zero (Operations Unit faults), BFLAOB enables the interface term.

7) $(BWF \cdot WI_2)$

This level holds off INTR if the CU is handling an ADBn or LDCF with DU modification until after the I Buffer has been filled.

9.3.1.4 Timing Diagram

Figure 9.3.1.4 is a general timing diagram of the important strobes within the Control Unit. The times shown represent the best case times with a one microsecond memory using $C as a reference.

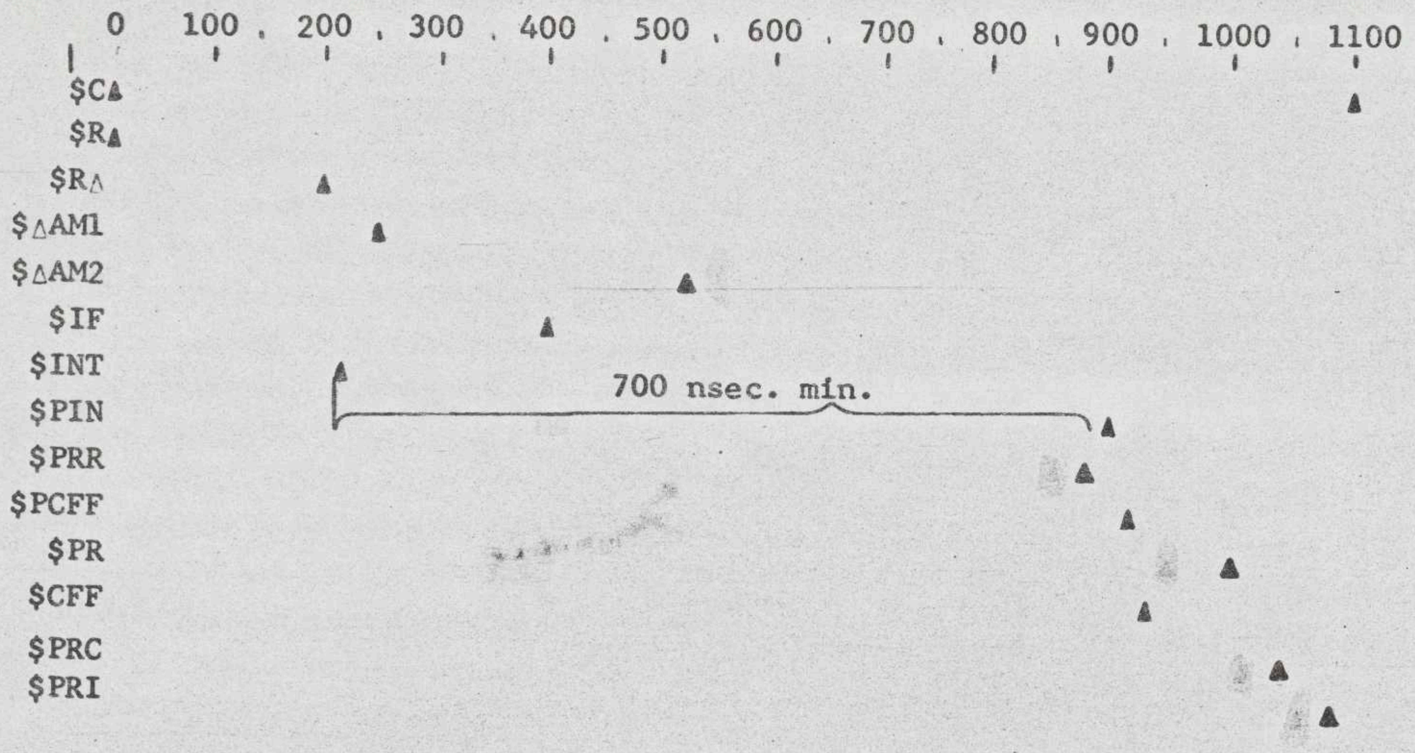


Figure 9.3.1.4 Major Control Unit Strobes, Timing Diagram

NOTES:

1) Times shown are for a one-microsecond memory

2) Processor is assumed to be initialized and is in a DIS state before starting a LDA, LDA, LDA, STA instruction sequence.

3) Processor is in Absolute mode with no address modification.

9.3.2 SEQUENCE CONTROL FLAGS

As mentioned previously in paragraph 9.2.1, the timing sequences within the Control Frame are determined by a number of sequence control flags. The conditions for changing the state of these flags is explained below.

9.3.2.1 Prepare Instruction (PI) (See 43D163387, sheet 31)

The PI flag set and reset equation shown below will be explained term by term. The PI flag is set as a function of $C and ¢ Set PI, and initialized by WF any time that a fault is about to be serviced.

$\text{¢Set PI} = \text{BSPJ1B} \cdot \overline{\text{DIS}} \cdot \overline{\text{PC}}$ (1)
$\text{BSPJ1B} = \text{PA} \cdot \overline{\text{TRGO}_1} \cdot \text{CT} \cdot \text{IC} \cdot \overline{\text{LSI}} \cdot \text{TXI}$ (2)
$+ \text{PI} \cdot \text{SPI}$ (3)
$+ \text{IC} \cdot \text{EA} \cdot \overline{\text{¢Set RT}} \cdot \overline{\text{LSI}} \cdot \overline{\text{TXI}} \cdot (\overline{\text{XED+XEC+RCU}}\text{+TERM+}$
$\overline{\text{FD+FT+FL+(LREG+SREG}}\text{)EA} \cdot \text{CT=3)}$ (4)
$+ \text{TRGO}_1 \cdot \text{EA}$ (5)
$+ \text{RCU} \cdot \text{EA} \cdot \text{ICTA}_{10}$ (6)
$+ \text{LSI} \cdot (\overline{\text{LREG} \cdot \text{¢Reset MSLF}}) \cdot \overline{\text{IC}} \cdot \text{EA}$ (7)

$\overline{\text{¢SetPI}}$ = when any of the terms in the above combined equation causes ¢Set PI to become false.

The meaning of each group of terms is as follows:

1) The Processor is not in a DIS (wait for interrupt) state or in the prepare cycle for an external interrupt.

2) In the PA cycle of a conditional transfer that is not going to transfer in the odd instruction register, and there is no load-store instruction combination or transfer in the I-register that would preclude going into a PI cycle to fetch a pair of instructions.

3) In a PI cycle and must have a second PI cycle (usually when doing a four-instruction fetch).

4) This equation represents the general case for going into PI; when finished with the odd instruction of a pair, pull another instruction pair to fill

---

9.3.2.1 - contd

the I-register as the I-register is moved to the Y- and C-working registers. The inhibit terms prevent going into PI for a RPT, RPD, RPL, RET, XEC or XED instruction, or a load-store instruction sequence or transfer instruction in the I-register.

5) A satisfied transfer with an effective address requires going into a PI cycle to pull new instructions to fill the I buffer.

6) These terms are for an RCU instruction which may change the state of the PI flip-flop to restore it to the condition it was when saved by a previous SCU instruction.

7) There was a load-store instruction combination in the I-register and the effective address for the instruction in the Y-register is available. This indicates that two instructions may be fetched to fill the I-register between the execution of the load instruction in the Y-register and the store instruction in the C-register.

9.3.2.2 Prepare Address (PA) (See 43D163387, sheet 31)

The PA flag set and reset equations are a bit complex and will be explained term by term. Set and reset terms on the equation are the same. If the combined equation produces a true output, the PA flag is set on the next $C received; when false, the flag is reset.

$$
\begin{aligned}
&\text{¢ Set } PA = BSPA1B \\
&BSPA1B = (\underline{BSPC1B\cdot}BSPD1B)+(WF\cdot\overline{PC}) \\
&BSPD1B = (\overline{RPD}\cdot\overline{DIS}) \underline{+ PI} && (1) \\
&BSPC1B = BSPE1B\cdot\overline{STOP}\cdot\overline{TRGO_1}\cdot PI\cdot\overline{PC} && (2) \\
&BSPE1B = [CT\cdot\underline{IC}] && (3) \\
&\qquad + [EA\cdot\overline{IC}\cdot\overline{FL}\cdot\overline{LSI}\cdot\overline{XED+XEC}\cdot\overline{RCU}] && (4) \\
&\qquad + [LSI\cdot IC\cdot EA] && (5) \\
&\qquad + [TXI\cdot EA] && (6) \\
&\qquad + [W\cdot\underline{FL}] && (7) \\
&\qquad + [FT\cdot\overline{TERM}\cdot\underline{EA}] && (8) \\
&\qquad + [\underline{EA}\cdot FD\cdot\overline{TERM}] && (9) \\
&\qquad + [\overline{IC}\cdot EA\cdot FL\cdot TERM] && (10)
\end{aligned}
$$

9.3.2.2 - contd

$$+ [IC \cdot RDE] \tag{11}$$
$$+ [(LREG+SREG) \cdot EA \cdot CT = 3] \tag{12}$$
$$+ [RCU \cdot EA \cdot ICTA_9] \tag{13}$$
$$+ [WI_1 \cdot (XEC+XED)] \tag{14}$$
$$+ [PI \cdot \overline{SPI} \cdot (\overline{XDE} + TRGO_1)] \tag{15}$$
$$+ [WC] \tag{16}$$

$\overline{\text{¢ Set PA}}$ = When any one of the terms in the above combined equation causes ¢Set PA to become false.

1&2) Currently in the prepare instruction address cycle for any instruction but a RPD or DIS, hte Processor is not stopped, not a satisfied conditional transfer, and not an execute interrupt cycle for an external interrupt.

3) A conditional transfer in the odd instruction register.

4) An effective address exists for the instruction in the even instruction register. This instruction is not an RPL, XED, XEC, or RCU and there is no load-store instruction combination in the I-register.

5) An effective address exists for the instruction in the odd location and there is a load-store instruction combination in the I-register.

6) There is a transfer instruction in the I-register and an effective address exists for the current instruction in the working registers.

7) Waiting for the operand from memory for an instruction being repeated under Operation Unit control of a RPL.

8) An effective address exists for an instruction being repeated under control of a RPT and no terminate condition exists.

9.3.2.2 - contd

9) An effective address exists for an instruction being repeated under control of a RPD and no terminate condition exists.

10) An effective address exists for an instruction being repeated under control of a RPL that is about to terminate, and it is in the even instruction register.

11) Working on the instruction in the odd instruction register but remember that a PI cycle was just completed for an XED instruction in the even instruction register.

12) Executing a LREG or SREG instruction except the last cycle.

13) An RCU instruction is being executed, which is going to restore the PA flag to its former status at the time that a previous fault condition, and subsequent SCU instruction, stored status.

14) Waiting for an instruction, or pair of instructions for the Y- and C-register that is the result of an XEC or XED instruction.

15) In a PI cycle for a satisfied transfer that was not a second PI cycle or the result of an XED instruction.

16) Waiting for the address of an external interrupt.

9.3.2.3 Prepare Address (PZ) (See 43D163387, sheet 35)

The terms for the set and reset of the PZ flag show that this flag is set whenever an RI, or IR modification is specified by the tag field of an instruction word. Setting of the PZ flag takes place with the $C that takes the Control Frame out of a WN (wait for indirect) cycle. A term for the RCU instruction allows PZ to reflect the state it had when stored by a previous SCU instruction.

3) Waiting for the indirect word for an instruction being repeated under control of RPT or RPD.

9.3.2.4 Prepare Return or Repeat Indirect (PN) (See 43D163387, sheet 35)

The PN flag is set under the following conditions as can be seen from the logic diagram:

1) When an effective address for a RTD instruction is present and no interrupt is present.

2) For a RET instruction with no interrupt present.

3) Waiting for the indirect word for an instruction being repeated under control of RPT or RPD.

4) Restoration of the state of the PN flag by an RCU instruction.

5) Whenever there is a fault on a "repeated" (RPT, RPD, RPL) instruction and the Control Frame must go into a PN cycle to terminate the "repeat."

9.3.2.5 Prepare IT Operand (PT) (See 43D163387, sheet 35)

The PT flag is set by the $C that takes the Control Frame out of the WN cycle; provided that an IT modifier was specified in the instruction or indirect word tag that designated I, or CI address modification; no RPT or RPD instructions being processed, and no interrupt present. The terms for the RCU instruction, as with the other important flags, effect the state of the PT flag depending on what was stored by a previous SCU instruction. The PT flag may also be enabled when in a PR cycle, since the normal sequence of prepare cycles for instructions with IT modifiers that cause a read-alter-rewrite command to the memory is PA→ PR→ PT.

9.3.2.6 Prepare Rewrite (PR) (See 43D163387, sheet 35)

The PR flag is set concurrently with the $C that ends a WN cycle for an indirect then tally word, where address modification involving a read-alter-rewrite memory cycle is required. Whenever the PR flag is set, the PT flag will be set on the next $C. However, PT set does not imply that PR was set. The IT modifications I, CI, ITS and ITB will not cause PR to set.

9.3.2.7 Prepare Execute Interrupt (PC) (43D163387, sheet 21)

PC is a level from any one of the eight flip-flops associated with the Interrupt Present Register (IPR). The IPR is covered in detail in paragraph 9.3.3.

9.3.2.8 Wait for Operand (W) (See 43D163387, sheet 33)

The W flag is set and reset as a function of the following equations:

$$W\ set = BWCF1B \cdot \$PIN$$

$$BWCF1B = (RCU \cdot GBO \cdot INC = 2) + PW\Delta + (RTD \cdot GBO \cdot INC = 0) \qquad (1)$$

where PW (43D163387, sheet 32) is the Pre-W flip-flop and is set by:

$$PWset = ¢Set\ PW \cdot \$R$$

$$¢Set\ PW = \overline{T} \cdot EA \cdot \overline{TERM} \cdot \overline{RSW} \cdot \overline{Y} \cdot \overline{OVSTINE} \cdot \overline{TRGO_2} \qquad (2)$$

$$\overline{OU\ STORE} \cdot \overline{TRGO_2}$$

1) The terms for RCU and RTD allow W to be set; enabling a strobe \$DRDY to be sent to the Operations Unit so the Indicator Register may be loaded as part of the RCU or RTD instruction execution.

2) This equation shows that PW, and subsequently W, will only be set for an Operations Unit operand from memory, since the term $\overline{T}$ is only true for Operations Unit instructions. The terms RSW and $\overline{Y}$ are for direct instructions that do not require memory accesses.

$$W\ reset = \overline{DPF\Delta} \cdot \$DA_2 + WF \qquad (3)$$

3) Indicates that W will be reset by the first data available pulse for single-precision memory operations and the second data available pulse for double-precision memory operations, or when a fault must be serviced.

9.3.2.9 Wait for Instruction 1 (WI_1) (See 43D163387, sheet 33)

As the equations below for WI_1 show, this flag is only set when fetching instructions to fill the Y- and C-registers. Normally two instructions are fetched and strobed into the I-register, then the contents of the I-register are transferred into the Y- and C-registers. The equations for WI_1 represents only those cases when instructions are fetched (that is, transfers, XEC, XED instructions and store-compare function) for the Y- and C-registers.

$$WI_1 \text{ set} = \$R\cdot \text{BWWI1B}$$

$$\text{BWWI1B} = \text{¢SetXPI} + (\text{PI}\cdot\text{XDE}) + \text{EA}\cdot(\text{XEC}+\text{XED}) \qquad (1)$$

$$\text{¢SetXPI} = \text{TRGO}_1\cdot\text{EA}+\text{SPI}\cdot\text{PI} \qquad (2)$$

$$\text{SPI} = \text{DPI} + \text{EXI}\cdot\overline{\text{XDE}} \qquad (3)$$

1) Extend the PI cycle (for a second pair of instructions for the T-register and subsequently the Y- and C-registers), or in the PI cycle for an XED instruction in the Y-register (requiring that the Y- and C-registers are to be reloaded) or there is an effective address for an XEC or XED instruction which also require that the Y- and C-registers be reloaded.

2) The term ¢Set XPI is present on any satisfied transfer with an effective address, or when in a PI cycle and a second PI cycle was entered.

3) A second PI cycle was entered any time that the base frame fault logic indicates a double PI cycle was accomplished, or a Control Unit store-compare (EX) function came up when no XED instruction was in the Y-register.

Summarizing briefly: The WI_1 flip-flop is set whenever the sequential execution of instructions requires the current and/or remaining instruction in the Y- and C-registers to be repulled from memory because of a transfer, XEC, or XED instruction or the EX (store-compare) function comes true. In addition, when the Processor is initialized and four instructions are fetched, the term DPI (equation 3) comes true from the fault

---

9.3.2.9 - contd

logic to cause WI_1 to set thereby allowing the first instruction pair to be strobed into the Y- and C-register directly.

$$WI_1 \text{ reset} = \$DA\cdot¢\text{Reset } WI_1$$

$$¢\text{Reset } WI_1 = DYC_\Delta\cdot\overline{DTC}_\Delta\cdot+\overline{DYC}_\Delta\cdot DTC_\Delta \qquad (4)$$

4) This indicates that WI_1 is reset whenever the data is brought into the Y- or C-register(s).

9.3.2.10 Wait for Instruction 2 (WI_2) (See 43D163387, sheet 33)

The equations for WI_2 represent the general case for a "normal" two instruction fetch: that is, WI_2 is set when waiting for an instruction pair to fill the I-register:

$$WI_2 \text{ set} = \$R\cdot PI\cdot\overline{SPI}\cdot(\overline{XDE}+\overline{TRGO_1}) \qquad (1)$$

$$+\ RCU\cdot INC{=}4\cdot\overline{ICTA_0}\cdot\overline{PDP}_\Delta\cdot\$PIN \qquad (2)$$

1) Any instruction fetch that is not the result of a second PI cycle, XED instruction in the Y-register, or a satisfied transfer.

2) Whenever an RCU instruction is recognized, a strobe is generated to set WI_2 if the original state of the PI flag was a "0" during a previous SCU instruction, since the two instructions saved by the SCU instruction must go into the I-register.

$$WI_2 \text{ reset} = \$DA[W_\Delta\cdot\overline{DPF}+\overline{DPS}]+WF+SFF \qquad (3)$$

3) This means that WI_2 is reset when data is on the lines to the Processor from the memory, that is the second half of a double-precision read-restore operation. (Both instruction words have been received from memory and reside in the I-register). The terms $\overline{DPF}$ and $\overline{DPS}$ are for a double-precision fetch and to inhibit the reset of WI_2 during the termination of "repeat" instruction.

9.3.2.11 Wait for Indirect (WN) (See 43D163387, sheet 33)

The WN cycle is entered whenever a previous PA, PT or PZ cycle does not yield an effective address for an operand, and no fault is being serviced.

$$\text{WNset} = \$C[(PA+PZ+PT)\cdot\overline{EA}\cdot\overline{WN}\cdot\overline{RR}\cdot\overline{WF}\cdot\overline{SNO}]$$

As the set equation shows, WN is set as soon as the fetch for the indirect word is started. The terms $\overline{RR}$, $\overline{WF}$ and $\overline{SNO}$ are to prevent the setting of the WN flip-flop for conditions that are going to cause the servicing of a detected fault.

$$\text{WNreset} = \$DA_2\cdot ¢\text{reset } WI_1 + WF$$

As the reset equation shows, WN is reset when data from memory is available to be strobed into the Y-or C-register. Level ¢reset WI_1 is high when data is to be brought into the Y- or C-registers. Whenever WF is present WN is forced reset.

9.3.2.12 Even/Odd Sequence (ICF, ICG) (See 43D163387, sheet 30)

The conditions for resetting and setting the IC (F,G) flag are shown in the equation below. Reset conditions are shown first since the Y-register (even instruction) is normally serviced before the C-register (odd instruction). IC (F,G) consists of two flip-flops due to loading requirements but with identical inputs.

$$\underline{\text{IC (F,G) reset}} = \$C\cdot ¢\text{Reset IC} + \text{PNLSW/ZDIRF}$$

$$\begin{aligned}
¢\text{Reset IC} &= [TXI\cdot IC_{\Delta} && (1)\\
&+ LSI\cdot EA\cdot IC && (2)\\
&+ EA\cdot IC\cdot FD\cdot\overline{TERM} && (3)\\
&+ RCU\cdot EA\cdot\overline{ICTA_5} && (4)\\
&+ \overline{LSI}\cdot\overline{ICTB_{17}}\cdot PI\cdot\overline{SPI}\ (\overline{XDE}+TRGO_1) && (5)\\
&+ WI_{1\Delta}\cdot XED]\cdot\overline{RDE}\cdot\overline{WF} && (6)
\end{aligned}$$

1) There is an unconditional transfer in the I-register and the Control Frame is currently working on the odd instruction in the C-register.

9.3.2.12 -contd

2) There is a load-stroe instruction in the I-register and an effective address exists for the instruction in the C-register.

3) The Control Frame has finished with the second instruction of the two instructions designated by an RPD instruction but has not terminated the RPD instruction sequence.

4) An RCU instruction is being executed which will restore the status of the IC (F,G) flag to a reset condition.

5) This group of terms represent the general case for resetting the IC (F,G) flag for the even instruction.

6) Waiting for the instruction pair designated by an XED instruction to come from memory. RDE and WF prevent the reset of IC (F,G). There was no XED instruction in the even instruction register. Had there been an XED instruction, IC must not change because the instructions being fetched as a result of the XED instruction, will come into the Y- and C-registers and the instruction in the Y-register will be prepared first. If the Processor is waiting to start a fault sequence IC (F,G) is not changed.

The conditions for setting IC (F,G) are as follows:

$$
\begin{aligned}
&\text{IC (F,G) set} = \$\text{C}\cdot\text{¢Set IC} && (7)\\
&\text{¢Set IC} = \{[\overline{(\text{LDR}+\text{STR})\cdot\text{EA}\cdot\text{CT}=3} && (7a)\\
&\qquad\cdot\overline{\text{DIS}} && (7b)\\
&\qquad\cdot\overline{\text{RPD}} && (7c)\\
&\qquad\cdot\overline{\text{¢Set RT}} && (7d)\\
&\qquad\cdot\overline{\text{STOP}}\cdot\overline{\text{TRGO}_1} && (7e)\\
&\qquad\cdot(\overline{\text{LSI}})\cdot[\text{CT}\cdot\overline{\text{IC}}+\text{EA}\cdot\overline{\text{IC}}(\text{TERM}+\overline{\text{FL}+\text{FT}})]] && (7f)\\
&\qquad+[\text{LSI}\cdot\text{PI}_{\Delta}\cdot\overline{\text{SPI}}] && (8)\\
&\qquad+[\text{RCU}\cdot\text{EA}\cdot\text{ICTA}_5] && (9)\\
&\qquad+[\overline{\text{IC}}_{\Delta}\cdot\text{ICTB}_{17}\cdot\text{PI}\cdot\overline{\text{SPI}}\ (\overline{\text{XDE}}+\text{TRGO}_1)]\} && (10)\\
&\qquad\cdot\overline{\text{WF}} && (11)
\end{aligned}
$$

9.3.2.12 - contd

7) The terms in this equation are the inhibits for those conditions where IC (F,G) is not to be set.

7a) Not doing a LREG or SREG instruction that is not complete.

7b) Not in a DIS state waiting for an external interrupt.

7c) Not currently preparing an RPD instruction in the even instruction location (RPD in the instruction register is not allowed).

7d) Not doing a RET or RTD instruction.

7e) The Processor is not stopped or recognizing a transfer.

7f) There is no load-store instruction combination in the I-register, no conditional transfer in the Y-register and an effective address and terminate condition exists for a RPT, RPD, RPL instruction in the Y-register; or an effective address exists for the instruction in the Y-register, and this instruction is not RPT or RPL.

8) The Control Frame is finishing a PI cycle that is not the second PI and there is a load-store instruction combination in the I-register.

9) An RCU instruction is being executed which may effect the setting of IC.

10) This equation represents the case for setting IC (F,G) for an odd instruction that if fetched in to the C-register.

11) Whenever a fault condition is about to be serviced IC (F,G) is not changed.

9.3.3 INTERRUPT PRESENT REGISTER (IPR) (See 43D163387, sheet 21)

The IPR is an eight-bit register, each bit of which corresponds to one of the eight (numbered 0-7) "ports" on the Processor. Each individual stage of the IPR is set as a function of an interrupt present from one of the ports connected to a Memory Controller, and a $C which initiates a PA, PZ, PN or PT address preparation cycle. When the Processor is in a DIS (delay until interrupt) state, the presence of an interrupt causes one of the stages of the IPR to be set for the port which causes the interrupt. Setting of one of the IPR stages causes a level designated PC to be generated which, when recognized by the fault logic, places the Processor in a prepare execute interrupt state.

9.3.3.1 Interrupt Sampling

Sampling of the interrupt present lines for each Processor port is accomplished with a $IPR which sets any stage (IPR_{0-7}) which has its interrupt present line (IP_{0-7}) high. Strobe $IPR is generated by a $C when the Processor is running and bit 2 of the current instruction a "0," by an interrupt present at one of the Processor ports when in the DIS state, by a Group VI Fault (Connect, Timer Runout, Shutdown), or by a Execute Fault set as a function of some of the Maintenance Panel switches.

$$\$IPR\ (\text{Processor running}) = \$C_{\wedge}\cdot\overline{MLSF}\cdot(PA+PZ+PN+PT) \qquad (1)$$
$$\cdot(MSY\cdot C28\cdot IC)+(MSY\cdot Y28\cdot\overline{IC})$$

1) Strobe $IPR is generated approximately 120 nanoseconds after the start of any PA, PZ, PN or PT address preparation cycle unless the instruction presently being executed has its interrupt inhibit bit set to a "1," or is a multiple operations unit cycle instruction (LREG, SREG, etc.).

$IPR (Processor in DIS state)

$$= (IP0+IP1\cdots+IP7+FRQ)\cdot DIS\cdot\overline{PA}(\overline{SPC}+\overline{SRF})\cdot\overline{PI} \qquad (2)$$

9.3.3.1 - contd

2) Strobe $IPR is generated whenever an interrupt present line is high from one of the eight memory ports or a Connect, Timer Runout or Shutdown Fault is detected. The term $\overline{PA}$ is to prevent interrupting if in the PA cycle of a DIS instruction rather than the Processor waiting in a DIS state. Both $\overline{SPC}$ and $\overline{SRF}$ in addition to $\overline{PI}$, are used to inhibit the generation of $IPR during certain areas of critical timing which could possibly allow $IPR during PI.

9.3.3.2 Reset Conditions

Resetting of the IPR is normally done whenever a $INT is sent to memory in response to an external interrupt, and when in a PC cycle. Strobe $IF is for cases where there will be no $INT issued as a result of a previous execute interrupt cycle.

If a terminate or stop-for-fault condition occurs, then the IPR is forced to reset and inhibit recognizing any interrupts until the particular condition ($\overline{TERM+WF\cdot SFF}$) serviced.

9.3.4 INPUT DATA CONTROL AND STROBE GENERATION FOR THE Y- C- AND I-REGISTERS

Bringing in data, in the form of instructions, to the Y-, C- and I-registers and how this data should be strobed into the effected registers is rather complex because of the four instruction look-ahead feature in the Control Frame. Generally speaking, the actions and conditions required to bring data into the Y-, C- and I-registers, fall into the following areas:

1) Control of the V (data-in) switch to control the data to the Y-, C- and I-register. Control of the VTA, VTB and TIC switches for data to the Y- and/or C-registers. Instructions can be brought in as a result of:

   a) A normal two- or four-instruction fetch.

9.3.4 - contd

b) Address modification on the instruction currently being processed.

c) Instructions which restore the Control Frame to some previous state (RTD, RCU).

d) Executing instructions which cause a repulling of new instructions, or a transfer of control (XEC, XED instructions and all satisfied transfers).

e) Fault conditions which cause a "snapshot" to be taken (the Y- and C-registers contents are strobed into the I-register prior to servicing the fault which caused the "snapshot").

2) Generation of strobe $YOC which is used to generate the actual strobes to strobe the Y- and C-registers. Strobe $YOC can be generated because of:

a) A "normal" instruction fetch resulting in instructions coming into the Y- or C-registers via the I-buffer or V-switch.

b) The ADB_n, LDCF, or RTD base instructions being executed.

c) Address modifications involving a read-alter-rewrite memory cycle (AD, SD, DI, DIC etc.)

d) An XEC or XED instruction in the Y- or C-register respectively, which will cause their contents to be replaced or restored.

3) Control of the strobes to the address, operation code (including bits up to bit 29), and tag portions of the instruction word that is brought in the Y and/or C-registers. The strobes being generated by $YOC.

4) Generation of strobes $ZIE and $ZIO to load the I-register whenever instructions are being brought in from memory as indicated by the condition of the WI_2 flag.

9.3.4 - contd

The four general conditions for control of data in the form of instructions to the Y-, C-, and I-registers and the resultant strobes generated will be examined in detail via the applicable equations.

9.3.4.1 Data-In-Control (See 43D163387, sheet 26)

There are eight control points that control the selection of data to the Y-, C- and I-registers: VTA, VTB (Y- and C-register); DTV (turn on ZDI_{0-29} for data from memory); BTTVOB (turn on ZDI_{30-35} for data from memory); BTV (turn on BS_{0-17} for instructions or address modification requiring the contents of the BS-adder); TIC for transferring data from the I- to Y- and C-registers; SN0, SN1 for transferring the Y- and C-registers to the I-register during the "snapshot" sequence for a fault. (Examine 43D160212 sheet 5 to see the relationship of the control points to data being brought into the Y-, C-, and I-registers.)

$VTA = [WF]$ (1)

$\quad + [(ADB_n + LDCF)\ \overline{CSR} \cdot \overline{IC} \cdot (DU+DL) \cdot \overline{CFR} \cdot \overline{WI_2}]$ (2)

$\quad + [PR \cdot (\underline{SCR} \cdot SD \cdot \underline{DI} \cdot DIC) \cdot \overline{IC}]$ (3)

$\quad + [DTC \cdot \overline{DYC} \cdot WN \cdot \overline{IC}]$ (4)

$\quad + [DYC \cdot BBVA1B]$ (5)

$BBVA1B = XED$ (5a)

$\quad + WC \cdot \overline{WF}$ (5b)

$\quad + \overline{ICTB_{17}} \cdot XPI$ (5c)

$\quad + \overline{IC} \cdot BYEJ1B$ (5d)

$BYEJ1B = XEC$ (5d-1)

$\quad + (\overline{XEC+XED}) \cdot RDE$ (5d-2)

$\quad + WN$ (5d-3)

$\quad + FL \cdot \overline{RDE} \cdot \overline{TERM}$ (5d-4)

$\quad + \underline{RET} \cdot RT$ (5d-5)

$\quad + \overline{DYC} \cdot IFR \cdot GBO \cdot ADB_n + LDCF + RTD)$ (5d-6)

1) Waiting for a fault condition where an XED instruction will be "brute-forced" into the Y-register.

2) Executing an ADB_n or LDCF instruction with bits 0-17 of the instruction word interpreted as direct

9.3.4.1 - contd

operand, but not an OU operand since the execution of the ADB_n and LDCF instructions take place in the Base and Control Frames. Had DU or DL address modification not been specified by the instruction word, a memory access would have to be made and the VTA level brought up for the "normal" case of data to the Y-or C-registers. (See equation 19 for details.). $\overline{CFR}$ and $\overline{WI_2}$ hold off the control until the operand is on the data in lines.

3) During the rewrite portion of a read-alter-rewrite memory cycle for the modifiers shown, the address portion of the indirect then tally word must be changed to the new modified address, prior to fetching an operand.

4) During ITS or ITB the second word of the indirect pair must be brought into the Y- or C-register. This equation accomplishes this.

5) Level DYC_Δ is the data to Y- or C-register flip-flop which is discussed in the next paragraph under strobe $YOC generation.

5a) An XED instruction is being executed which will bring in two new instructions to replace the ones currently in the Y- and C-registers.

5b) The Control Frame is in the wait cycle of an execute interrupt and is waiting for the interrupt block address to come back from the highest unmasked interrupt cell in the memory controller. Level $\overline{WF}$ is to ensure that the previous "snapshot" and fault sequence is complete before entering WC.

5c) The first pair of instructions for a four-instruction fetch is about to come into the I-register and the Y-register must be loaded with the even instruction word.

9.3.4.1 - contd

5d-1) Executing an XEC instruction which required pulling a new instruction into the Y-register.

5d-2) Repulling the old instruction that formally occupied the C-register but was lost due to an XED instruction in the Y-register. An XEC or XED instruction is not allowed as the first instruction of a pair of instructions specified by previous XEC or XED. Notice that the instruction coming in will come into the C-register via VTB as indicated by RDE and the fact that IC is set for this condition (see equation 7 of paragraph 9.3.2.11).

5d-3) Waiting for an indirect word from memory.

5d-4) Executing the repeated instruction specified originally by an RPL instruction, no terminate condition has been detected, and the repeated instruction was not an XED in the even instruction location.

5d-5) Executing a RET or RTD instruction when in a general base operation state (RT contains RTD·GBO).

5d-6) During ADB_p or LDCF or RTD the operand is brought into the Y- or C-register and then added to or sent to the proper register in the Base Frame.

The equations for VTB are the same as VTA except that IC takes the place of $\overline{IC}$, $ICTB_{17}$ takes the place of $\overline{ICTB_{17}}$, and the equation of (5) does not contain XED (5a), because an XED is not allowed in the odd instruction location (C-register).

9.3.4.1 - contd

DTV (turn on ZDI_{0-29}) = BBVP1B (6)
+ BDVTOB (7)
+ RCU·GBO (8)
+ PR·AD (9)
+ WI_2 (10)

6&7) These equations are the same as (5) above and depend on whether the Y- or C-register must be loaded.

8) Execution of an RCU instruction and restoring the Control Frame status that was stored by a previous SCU instruction.

9) Performing an IT type AD address modification, where the delta portion of the IT word from memory must be on the ZDI lines to be added to the operand address in the YS-adder (DTV will in turn allow BTTVOB and ZDI_{30-35}).

10) In the normal wait period for an instruction pair to come from memory to fill the I-register

BTTVOB = DTV (11)
+ PR·SD (12)

11) Any time ZDI_{0-29} is turned on as a result of the previous condition shown in equations 1-10, all 36 bits on ZDI must be present.

12) During IT type SD address modification, only the delta portion of the indirect then tally word is necessary to form the new operand address that is rewritten into memory during the PR cycle.

BTV (BS_{0-17} to ZDI) = Equation (2), (3) and (4) above (keeping in mind that IC differs for VTA and VTB). The BS-adder is switched to the input of the Y- and C-register via the VTA, VTB switches during the PR cycle for indirect then tally address modification. At the end of the PR cycle the new modified address is strobed into the Y- or C-register in preparation for the final operand addressscycle.

9.3.4.1 - contd

TIC (Transfer I to the Y- and C-registers, see 43D163387 sheet 24)

$$
\begin{array}{lr}
= [\mathrm{TXI} & (13) \\
\quad + \mathrm{LSI} & (14) \\
\quad + \mathrm{PI} \cdot \overline{\mathrm{SPI}} \cdot (\overline{\mathrm{XDE}} + \mathrm{TRGO}_1)] & (15) \\
\quad \cdot \mathrm{TIB} & (16) \\
\quad \cdot [\overline{\mathrm{DYC}_\Delta + (\mathrm{DTC}_\Delta (\mathrm{ITS} + \mathrm{ITB} + \mathrm{WN}))}] & (17)
\end{array}
$$

13) If there is a transfer instruction in the I-register no PI cycle is entered to fetch a new pair of instructions to fill the I-register, so TXI is needed to generate TIC.

14) When a load-store instruction combination is in the I-register, a PI cycle is postponed until the time interval between the load and store instruction. LSI allows the I-register to be strobed into the Y- and C-registers without entering a PI cycle.

15) This group of terms represent the general case for fetching a pair of instructions to fill the I-register and represents a normal PI cycle for the second fetch of a four instruction fetch.

16) If TIB is not set indicating that the contents of the I-register are ready to be transferred, TIC is inhibited.

17) If data is to be brought into the Y- or C-register, then the I-register is not to be transferred, and TIC is inhibited.

SNO, SNI (transfer Y- and C-registers to I-register) - These control points are brought up by the fault logic in the Base Frame. Discussion of the conditions that generate SNO, SNI are covered in Section 10.5.

9.3.4.2 Strobe $YOC Generation (See 43D163387 sheet 24)

Strobe $YOC is generated any time that data is brought into the Y- or C-register. Control of the strobe, to the various bits of the registers, is covered in the next paragraph. As the equation below shows, generation of $YOC is possible under three conditions: data coming into the Y- or to C-registers as part of a normal two, or four-instruction fetch; changing of existing data due to instruction or address modifications that involve a read-alter-rewrite memory cycle; anytime an ADB_n or LDCF instruction with a DU or DL (direct operand) modifier is encountered and no fault condition exists:

$$
\begin{aligned}
& \quad \$R \\
\$YOC = & + \$GDA \\
& + \$DS \cdot PR && (18) \\
& + \$PR \cdot (ADB_n + LDCF) \cdot \overline{CSR} \cdot (DV + DL) && (19)
\end{aligned}
$$

where:

$$\$GDA = \$DA_2 \cdot (DYC_{\Delta 2} + (DTC_{\Delta} \cdot (ITS + ITB + \overline{WN}))) \qquad (20)$$

18) The Control Frame has stored the modified data for a read-alter-rewrite memory cycle, that is the result of some IT address modification, into the DOR register in the Interface Frame. Now the final operand address preparation cycle may be entered. Strobe $YOC is generated because a modified address may be strobed into the applicable instruction register as a result of the specified address modification, and this address used to form the effective address for the operand.

19) If the ADB_n or LDCF instructions have direct operands, then no memory access is required. The adding that takes place in executing ADB_n or LDCF is to the address or control field of the specified address base register(s); the BS-adder does the adding, then BS_{0-17} is sent to the Base Frame. Since the possibility exists for an internal base to be added to the information brought into the BS-adder on a direct operand (if bit 29 of the instruction word is a "1" and an internal base is specified) two operations must be completed:

   a) Add the internal base in the BS-adder with the direct operand contained in the instruction word.

9.3.4.2 - contd

b) For ADB_n, add the specified ABR in the BS-adder with the contents of the direct operand and internal base. For the LDCF instruction the internal base addition effectively destroys the ABR control field information.

Since both the internal base and ABR designated by the instruction are added in the BS-adder (for ADB_n); the intermediate results of the first addition (internal base and direct operand) must be saved in the applicable instruction register (Y or C). Hence, generation of \$YOC in this case. When preparing to enter a fault routine \$YOC is inhibited.

20) This group of terms represent the general case for generating a strobe \$YOC for the Y- or C-register during a normal instruction fetch; in cases where an ITS or ITB tag modifier is encountered as a result of address modification; or when the C-register must be loaded independently of the Y-register (XED instruction in Y-register, or XEC instruction in C-register).

In the equation of (20), the terms $DYC_{\Delta 2}$ and DTC_{Δ} indicate when data is to be brought into the Y- or C-registers. Referring to the logic diagram for the inputs to DYC, note that DYC is set any time the Control Frame is waiting for an instruction or indirect word to fill the Y- and/or C-register; specifically:

a) The operand for the base instructions which are executed within the Control Frame (ADB_n, LDCF, RTD).

b) Instructions as a result of a WI_1 cycle which is to result in an eventual two-instruction fetch for the Y- and C-registers.

9.3.4.2 - contd

This represents the general case where the I-register is to be strobed into the Y- and C-register as two new instructions are brought into the I-register, or when an indirect word is brought in as a result of address modification.

c) Any prepare cycle (except PR) which may cause new data to be brought into the Y- or C-register.

An examination of the inputs for DTC shows that DTC is only set if DYC is set. What this indicates is that DYC set is for the first half of a double-precision instruction or indirect word fetch to the Y- or C-register. DTC is for the second half of a double-precision instruction or indirect word (ITS or ITB modification) fetch.

9.3.4.3 Strobe Control for Y- and C-Registers (See 43D163387 sheet 25)

Strobe \$YOC is generated whenever data is brought into the Y- or C-registers as shown in equations (18) through (20). Control of strobe \$YOC to the major portions of the Y- or C-register is divided into three parts: \$ZYE and \$ZYO for bits 0-17; \$ZCE and \$ZCO for bits 18-29; \$ZCF and \$ZCG for bits 30-35. The conditions which control the distribution of the strobe to the Y-register is essentially the same as to the C-register, so only distribution to the Y-register is discussed.

$$\begin{aligned} \$\mathrm{ZYE}_{0-17} &= \$\mathrm{YOC}\cdot(\overline{\mathrm{EXI}}\cdot \mathrm{BYSC1B}) && (21)\\ \mathrm{BYSC1B} &= \mathrm{VTA}\cdot\overline{\mathrm{BVT}}+\mathrm{VTA}[(\underline{\mathrm{SD+}}\mathrm{DI+DIC+SCR})+(\mathrm{ADB}_n+ \\ &\quad \mathrm{LDCF}\cdot(\mathrm{DU+DC})\cdot\overline{\mathrm{CSR}})] && (21a)\\ &\quad + \mathrm{YOE} && (21a\text{-}1)\\ &\quad + \mathrm{TIC} && (21a\text{-}2)\end{aligned}$$

21) The address field (bits 0-17) is strobed any time data is brought into the Y-register, the store-compare function is not present (Y-register only) and the terms in 21 through 21a-2 are satisfied.

21a) The two groups of terms with VTA are for conditions where the address field of the instruction word is under IT type SD, DI, DIC, or SCR address modification which requires a new updated operand address.

9.3.4.3 - contd

The term $\overline{BVT}$ contains both $\overline{IC}$ and PR in addition to SD, DI, DIC, and SCR, thereby allowing the new modified address of the indirect then tally word to be strobed into Y_{0-17}, and when the operand for a ADB_n or LDCF with DU or DL modification must be strobed into Y.

21a-1) The term YOE is for conditions where a store-compare function is detected or a fault condition is being serviced which results in Y_{0-17} being reloaded.

21a-2) TIC represents the general case where the Y-register is going to be loaded from the I-register for the next pair of instructions in sequence.

$$\$ZCE_{18-29} = \$YOC \cdot (\overline{EX_1} \cdot (\overline{IOS} + PI) \cdot BCSY1B) \qquad (22)$$

$$BCSY1B = VTA \cdot \overline{BVT} + YOE + TIC \qquad (22a)$$

22) This equation shows that the operation code portion of the word that is brought into the Y-register is only strobed when a new instruction is fetched. Terms $\overline{IOS}$ or PI also include provisions for the instructions that, in effect, bring an operand into the Y-register for execution within the Control Frame (instructions ADB_n, RTCD, LDCF, RPL, RPT, RPD, RET). The "repeat" instructions are included since the repeated instruction(s) that is designated, does not have the operation code changed during the repeat process.

22a) This equation represents the general group of conditions for which the operation code is changed when data is brought into the Y-register.

$$\$ZCF_{30-35} = \$YOC \cdot [(\overline{ITS} + PI)(\overline{EX_1})(BCYT1B)] \qquad (23)$$

23) The terms for this equation have been explained before with the exception of $\overline{ITS}$. By reference to sheet 26 for the generation of $\overline{ITS}$, note that if a "repeat" or a "return" type is being executed, then ITS is generated and strobe $\$ZCF_{30-35}$ is inhibited.

9.3.4.3 - contd

This is because of restrictions on the use of the address modification that can be designated by the tag field of these instructions (RET, RTCD, RPT, RPD, RPL).

Referring back to sheet 25, note that control of the strobes for the C-register is identical to the Y-register except for the term $\overline{EX_1}\cdot\overline{EX_1}$ is required as an inhibit on the Y-register strobes whenever a store-compare (EX) function is generated against the C-register. The C-register must be reloaded by the EX function forcing an XEC instruction to repull the instruction in the C-register. Since EX_1 is required to generate YOE and enable the strobe control to both the Y- and C-registers, $\overline{EX_1}$ is required to prevent the Y-register from getting strobed and losing its contents when the C-register is reloaded.

9.3.4.4 Strobe Generation for I-Register (See 43D163387, sheet 33)

Information is strobed into the I-register under two conditions:

1) Doing a "snapshot" sequence where the Y- and C-registers are loaded into the I-register in preparation for entering a fault routine.

2) Normal instruction fetches during a WI_2 cycle for the two instruction pair.

Strobes $ZIE, and $ZIO are generated for the two general conditions shown, as can be seen from the logic diagram.

9.3.5 ICT STROBES AND TRACKING LOGIC FOR ICTC, ICTB, ICTA' AND ICTA INSTRUCTION COUNTERS

The Control Unit contains four instruction counters to hold the address of an instruction during the various stages of address preparation or execution within the Control Unit and when being executed within the Operations or Appending Unit:

ICTB - contains the address of the even instruction of the instruction pair contained in the I-register.

9.3.5 - contd

ICTC - contains the address of the even instruction of the pair of instructions in the Y- and C-registers during the course of a normal instruction/operand address preparation sequence. When the Control Unit is restored as a result of an RCU instruction from a previous fault condition, or if a transfer instruction is encountered, ICTC could contain the address of the odd instruction.

ICTA' - contains the address of the even or odd instruction which in the Y- or C-register is to be sent to the Operations Unit (for operations unit instructions) or executed within the Control or Base Frames (control frame or base frame instructions). When the Control Frame is restored as a result of an RCU instruction from a previous fault condition, ICTA' is used to hold the bits that will restore the various status flags within the Control Frame (with the exception of the "repeat" flags which are restored from the ZDI lines as data is brought from memory).

ICTA - contains the address of the even or odd instruction sent to the Operations Unit for execution (for operation unit instructions) or the address of nonoperations unit instructions executed within the Control and Base Frame.

The data paths between the four ICT counters are shown on 43D160212 sheet 5. Notice that bits 0-16 of ICTB are transferred progressively to ICTC to ICTA' to ICTA. Bit 17 of the ICTA' and ICTA counters may be changed depending on whether a normal instruction sequence for preparation and execution is followed, or a fault of some type occurs. The blocks labelled "V" and "ATA" perform the logical functions to decide whether bit 17 of ICTA' and ICTA should be "0" or "1." Generally speaking, the ICT tracking falls into three distinct categories:

1) Cases where instructions are being fetched, prepared, and executed in a normal fashion (in sequence, no transfers, faults, etc.).

9.3.5 - contd

2) Cases where a fault is detected and the ICTC register must be saved for nonoperations unit (except for parity) faults, or ICTA saved for operations unit faults, and parity faults for nonoperation unit instructions.

3) When an RCU instruction is encountered to restore the status of the Processor, certain of the words specified by the RCU instruction are used to reset or set the Control Frame flags from ICTA' and restore the ICTC counter.

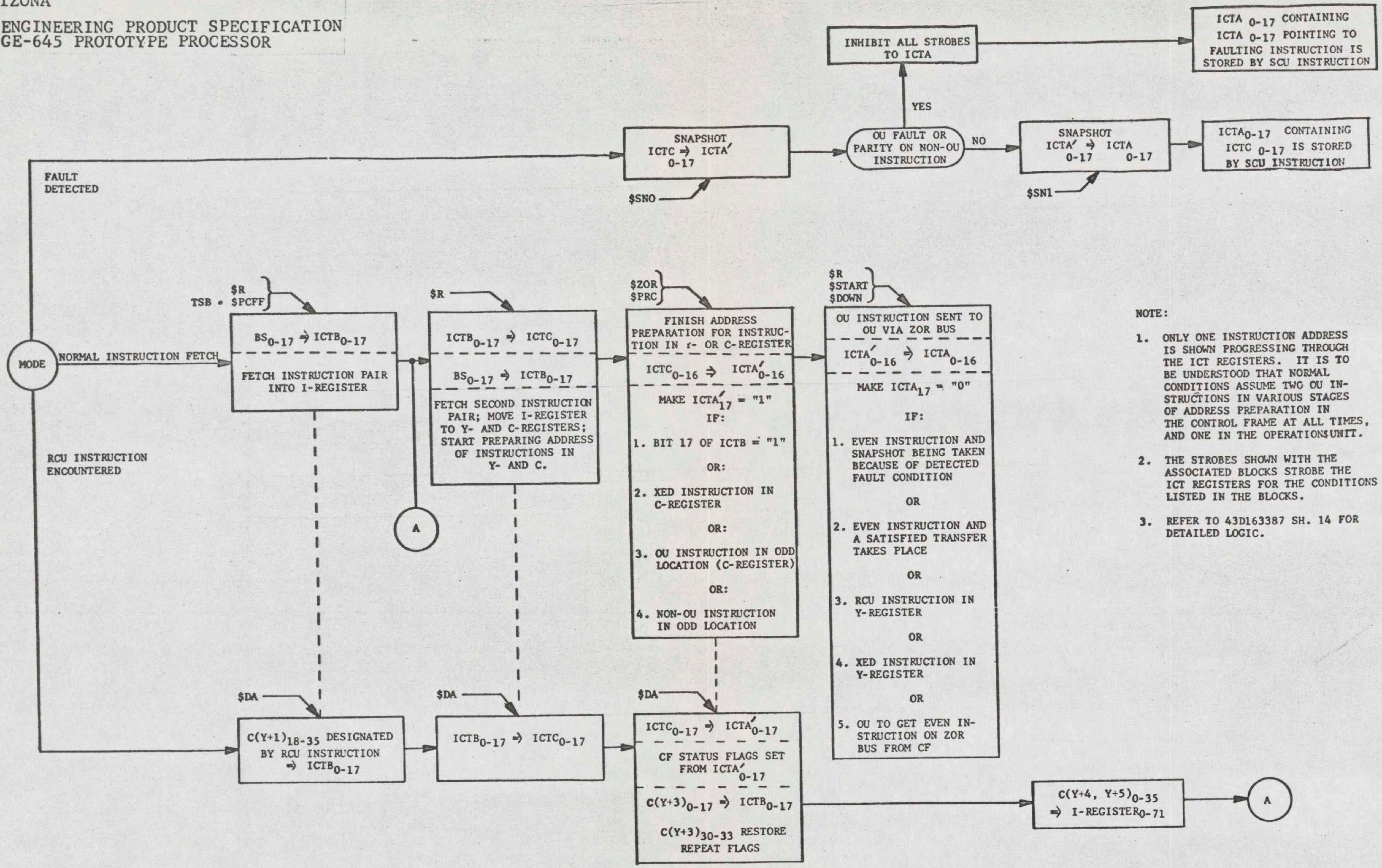
The three different modes are illustrated in Fig. 9.3.5.5 and the logic diagram is 43D163387, sheet 14.

9.3.5.1 Normal ICT Tracking

Normally, instructions are fetched in pairs by switching the ICTB register to the YS-adder input adding +1 in both the YS- and BS-adders and fetching a pair of instructions for the I-register.

(Refer back to the paragraph and flowchart on instruction fetches for details.) When a $R is generated to produce an interrupt to memory for the pair of instructions for the I-register, ICTB$_{0-17}$ is strobed with the address of the instruction pair still contained in the BS-adder. The same process is repeated for another instruction pair, the instructions in the I-register are moved to the Y- and C-registers, and ICTC$_{0-17}$ is strobed with the contents of ICTB$_{0-17}$. Address preparation is then started for the instruction in the Y-register.

When the instruction in the Y-register is ready for execution, and it is an operations unit instruction, the operation code is stored in the RE buffer within the Control Frame until the Operations Unit can accept it. Concurrent with saving the operation code in the RE buffer, a strobe $ZOR that strobes the RE buffer transfers ICTB$_{0-16}$ to ICTA'$_{0-16}$. Bit 17 of ICTA' is set


MODE
NORMAL INSTRUCTION FETCH
FAULT DETECTED
RCU INSTRUCTION ENCOUNTERED
TSB • $R $PCFF
BS0-17 ⇒ ICTB0-17
FETCH INSTRUCTION PAIR INTO I-REGISTER
$R
ICTB0-17 ⇒ ICTC0-17
BS0-17 ⇒ ICTB0-17
FETCH SECOND INSTRUCTION PAIR; MOVE I-REGISTER TO Y- AND C-REGISTERS; START PREPARING ADDRESS OF INSTRUCTIONS IN Y- AND C.
A
$ZOR $PRC
FINISH ADDRESS PREPARATION FOR INSTRUCTION IN Y- OR C-REGISTER
ICTC0-16 ⇒ ICTA′0-16
MAKE ICTA′17 = "1"
IF:
1. BIT 17 OF ICTB = "1"
OR:
2. XED INSTRUCTION IN C-REGISTER
OR:
3. OU INSTRUCTION IN ODD LOCATION (C-REGISTER)
OR:
4. NON-OU INSTRUCTION IN ODD LOCATION
$R $START $DOWN
OU INSTRUCTION SENT TO OU VIA ZOR BUS
ICTA′0-16 ⇒ ICTA0-16
MAKE ICTA17 = "0"
IF:
1. EVEN INSTRUCTION AND SNAPSHOT BEING TAKEN BECAUSE OF DETECTED FAULT CONDITION
OR
2. EVEN INSTRUCTION AND A SATISFIED TRANSFER TAKES PLACE
OR
3. RCU INSTRUCTION IN Y-REGISTER
OR
4. XED INSTRUCTION IN Y-REGISTER
OR
5. OU TO GET EVEN INSTRUCTION ON ZOR BUS FROM CF
SNAPSHOT ICTC ⇒ ICTA′ 0-17
$SNO
OU FAULT OR PARITY ON NON-OU INSTRUCTION
YES
NO
INHIBIT ALL STROBES TO ICTA
ICTA 0-17 CONTAINING ICTA 0-17 POINTING TO FAULTING INSTRUCTION IS STORED BY SCU INSTRUCTION
SNAPSHOT ICTA′ ⇒ ICTA 0-17 0-17
$SN1
ICTA0-17 CONTAINING ICTC 0-17 IS STORED BY SCU INSTRUCTION
$DA
C(Y+1)18-35 DESIGNATED BY RCU INSTRUCTION ⇒ ICTB0-17
$DA
ICTB0-17 ⇒ ICTC0-17
$DA
ICTC0-17 ⇒ ICTA′0-17
CF STATUS FLAGS SET FROM ICTA′0-17
C(Y+3)0-17 ⇒ ICTB0-17
C(Y+3)30-33 RESTORE REPEAT FLAGS
C(Y+4, Y+5)0-35 ⇒ I-REGISTER0-71
A

NOTE:

1. ONLY ONE INSTRUCTION ADDRESS IS SHOWN PROGRESSING THROUGH THE ICT REGISTERS. IT IS TO BE UNDERSTOOD THAT NORMAL CONDITIONS ASSUME TWO OU INSTRUCTIONS IN VARIOUS STAGES OF ADDRESS PREPARATION IN THE CONTROL FRAME AT ALL TIMES, AND ONE IN THE OPERATIONS UNIT.

2. THE STROBES SHOWN WITH THE ASSOCIATED BLOCKS STROBE THE ICT REGISTERS FOR THE CONDITIONS LISTED IN THE BLOCKS.

3. REFER TO 43D163387 SH. 14 FOR DETAILED LOGIC.

FIG. 9.3.5 GENERALIZED ICT TRACKING, FLOWCHART

9.3.5.1 - contd

to a "1" or remains a "0" for those conditions as shown on Fig. 9.3.5. For the particular case we are discussing ICTA'$_{17}$ = "0."

When the Operations Unit is ready to accept the operation code and begin executing the instruction, a strobe $START is generated and ICTA'$_{0-16}$ is transferred to ICTA$_{0-16}$. Bit 17 of ICTA will be a "0" because of the conditions shown on Fig. 9.3.5. This process is repeated as long as operations unit instructions are being fetched, prepared and executed in the normal manner. The timing between the various strobes varies greatly with the instructions fetched, operand address modification, execution, etc. The tracking of bit 17 when transferring an operations unit instruction address from ICTA' to ICTA is controlled by the B' flip-flop shown on sheet 14. This flip-flop is only set when the Operations Unit is ready to accept an instruction not when it is put on the ZOR bus to the Operations Unit. In a sense, the B' flip-flop tracks the B control flip-flop for the ZOR bus switching of the RE, RO buffers to the Operations Unit.

For nonoperations unit instructions, ICTA' and ICTA tracking is the same except that the address is transferred from ICTC to ICTA' by $PRC and from ICTA' to ICTA by $R; $R to ICTA implying that the nonoperations unit instructions have already been executed.

9.3.5.2 Fault Detected Tracking

If a fault condition is detected by the processor fault logic in the Base Frame, a "snapshot" sequence is initiated by the Base Frame. Strobe $SN0 strobes ICTC$_{0-17}$ into ICTA'$_{0-17}$. The conditions determining whether ICTA'$_{17}$ is a "0 or "1" are the same as the normal case. When $SN1, the second snapshot strobe is received, ICTA'$_{0-17}$ is transferred to ICTA$_{0-17}$ subject to the same conditions as shown for normal tracking of ICTA$_{17}$. However,

9.3.5.2 - contd

if the fault detected was an operations unit fault (divide check, overflow, parity error in operand), or parity on a nonoperations unit instruction operand, the original $ICTA_{0-17}$ is not changed. This means that when the expected SCU instruction in the processor fault vector is encountered, $ICTC_{0-17}$ is stored for nonoperations unit faults except parity, or $ICTA_{0-17}$ for operations unit faults and nonoperations unit parity faults.

9.3.5.3 RCU Instruction Tracking

When an RCU instruction is encountered, the Control Frame is restored to its status prior to the condition which caused some fault. All the status flags (PA, PI, PZ, etc.) are restored from the $ICTA'_{0-17}$ counter. The "repeat" flags (FT, FD, FL, RF) are restored directly from the ZDI data input lines at the same time that the ICTB counter is loaded via the ICT switch. When the sixth word designated by the RCU instruction is brought into the I-register, bits 36-71, a normal two-instruction fetch takes place in the Control Frame moving $ICTB_{0-17}$ to $ICTC_{0-17}$ and the I-register to the Y- and C-registers.

9.3.6 CT-REGISTER (See 43D163387 sheet 29)

The CT-register is a six bit register used to hold the tag field (bits 30-35) of the instruction or indirect word for IR and IT address modification, and the delta field for "repeat" instructions. For R-type address modification, the CT-register contains the tag field of the instruction word, but this field is not used.

Data for the CT-register comes from the ZI bus for normal address preparation and from the ICTA' register during restore operations resulting from an RCU instruction. Use of the the tag field is as specified for the particular address modification employed.

Whenever the "repeat" instructions are encountered, the delta field is strobed into the CT-register at the

9.3.6 - contd

end of the PA cycle of the "repeat" instruction. When the repeated instruction(s) is being executed, only R and RI modification is permitted and the tag is not strobed into the CT-register because it will destroy the delta field required for the "repeat" operation. At the time of termination for the repeated instruction(s), the delta field is forced to look like an R modifier. This is done by resetting CT_0 (bit 30 of the tag field of an instruction or indirect word) and preventing the possibility of an erroneous IR tag which could cause further action. Refer to the logic diagram for the exact terms which control the strobe and data.

9.3.7 RE-RO-REGISTERS; $ORY, $ZOR AND B CONTROL (See 43D163387 sheet 27)

Each time the Control Frame sends an operation code to the Operations Unit, it must send a strobe $ORY indicating that the operation code is ready at the input to the O-register in the Operations Unit. The Operations Unit replies with a strobe $YRY when the instruction has been decoded, and the operand is presented to the Operations Unit. The Control Frame then knows that another instruction may be sent; provided certain other conditions are met. The dispatching of an even or odd operation code, to the Operations Unit, is controlled by the "B" flip-flop in the Control Frame; if "B" is reset, an even instruction is sent, if set, an odd instruction is sent.

Because four instructions can be contained in the Control Frame, the possibility exists (and will generally be the rule rather than the exception), that the Control Frame is one or two instructions ahead of the Operations Unit and has an instruction ready before the Operations Unit can accept it. Therefore, the Control Frame is provided with two registers (RE and RO) to buffer the even and odd operation codes. These registers hold the nine-bit operation code that is to be put on the ZOR bus to the Register Frame of the Operations Unit. Which register is selected (RE or RO) depends on the state of the "B" flip-flop. As soon as the Operation Unit can accept an operation code, it sends back strobe $YRY to the

9.3.7 - contd

Control Frame allowing the "B" flip-flop to switch the other operation code on the ZOR bus. The "normal" sequence is to load both the RE and RO buffers and send the even operation oode to the Operations Unit. When the Operations Unit sends back $YRY, the Control Frame sends the odd operation code. The RE and RO buffers are then reloaded from the next instruction pair in the Y- and C-registers. The Operations Unit is allowed to get no more than three instructions behind the Control Frame under any conditions because of inhibits on the CU/OU interface term equation for $C. The only deviation from the "normal" sequence, is when there is a nonoperations unit instruction in the even instruction location. In this case, the RE and RO buffers will not get loaded with the even instruction, but a flag is set to indicate that the even instruction was a nonoperations unit instruction. When the prepare cycle for the odd instruction is in progress, and provided it is an operation unit instruction, then the RO buffer will be loaded at the same time that the "B" flip-flop switches the RO buffer output to the ZOR bus.

If the nonoperations unit instruction was in the RO (odd) buffer, then the "B" flip-flop is not toggled switching RO to the ZOR bus.

Concurrent with the switching of the RE or RO buffer outputs to the Operations Unit, $ORY is also generated by the Control Frame to indicate that an operation code is on the ZOR lines. Strobe $ORY is generated from the same strobe that generates $ZOR BUFFER, but is delayed 108 nanoseconds after $ZOR BUFFER, to enable the data on the ZOR bus lines to settle. A $ZORΔ is also generated to strobe the ICTA' register approximately 90 nanoseconds after $ZOR.

9.3.7.1 Status Flag

Whether or not the RE and RO buffers will be strobed with a new operation code, and when the "B" flip-flop is toggled, is defined by the state of five status flags:

1) ONR -- Operations Unit Not Ready

9.3.7.1 - contd

2) EOC -- Even Operation Code

3) OOC -- Odd Operation Code

4) EOF -- Even Operation Code First

5) NOS -- Not Operations Unit Sequence

These status flags are explained in reference to their set and reset equations as follows:

1) ONR set = \$POC
reset = \$YRY+XF+WF+TRMF

Strobe \$POC is a strobe that precedes \$ORY and is the result of satisfying all conditions to issue \$ORY to the Operations Unit. When the ONR flag is set, the B flip-flop state cannot be changed, and additional \$ZOR and \$ORY strobes cannot be generated. This condition will remain until the Operations Unit is not busy; indicated by a \$YRY which resets the ONR flag. Strobe \$YRY is only received when the Operations Unit is able to accept a new operation code.

2) EOC set = $\$Z\cdot(EA\cdot\overline{IC})$
reset = $\$ORY\cdot\overline{B}+WF$

The terms $EA\cdot\overline{IC}$ indicate that an effective address for an operand, for the instruction in the Y-register, is available from the BS-adder. Strobe \$Z is defined as follows:

$$\$Z = \$PRR\cdot[TRMF\cdot\overline{Y}\cdot\overline{ROS}\cdot(T+PN(\underline{R}ET+RTD))] + \$_{\Delta}AM2\ [\overline{TRMF}\cdot TSX_n+ROS+Y\cdot\overline{T}]$$

The terms associated with \$PRR represent the conditions for slow operations unit instructions and allow \$Z to be generated approximately 340 nanoseconds later than for fast operations unit instructions. Inhibit terms are for: not terminating a "repeat" instruction ($\overline{TRMF}$); not a direct operations unit operand ($\overline{Y}$); not operations unit stores or read-alter-rewrite type instructions ($\overline{ROS}$). The RET and RTD instructions are executed in part within

9.3.7.1 - contd

the Control Unit while the T term defines nonoperations unit instructions

All terms associated with $\$_{\wedge}AM2$ are for conditions which represent fast operations unit instructions and consequently require less time for decodes to settle and operations to be executed. Transfer and set index register (TSX_n) is a special case transfer that is treated as a fast operations unit instruction.

Resetting of the EOC flag depends on $\$ORY$ and $\overline{B}$; indicating that whenever the Operation Unit is sent a strobe indicating that an operation code from the Y-register is on the ZOR bus, that EOC will be reset.

3) $OOC\ set = \$Z\ (EA \cdot IC)$
   $reset = \$ORY \cdot B + WF$

The conditions for the set and reset of the OOC flag is identical to the EOC flag except for the term IC and B.

When IC is true, the operation code is from the C-register. When B is true, the odd operation code has been sent.

4) $EOF\ set = EOC \cdot \overline{OOC}$
   $reset = \$ORY \cdot \overline{B} + WF$

This means that EOF is set when an even operation code from the Y-register is to be sent to the Operations Unit, before the odd operation code. The only time that EOF will not be set is if the even operation code was not an operations unit instruction, and EOC did not get set.

5) $NOS\ set = \$R \cdot [PI \cdot XDE + (EA + \overline{TRGO} \cdot CT)$
   $(T + PN(RET + RTD))]$
   $reset = \$ORY + WF$

The group of terms for the set equation for the NOS flag shows that NOS is set for all nonoperations unit instructions; or portions of the execution of some instructions that involve the Operations Unit and Control Unit.

9.3.7.2 B Flip-Flop Control, $ZOR Buffer and $ORY Generation

The signal levels that control the state of the B flip-flop also generate a pre-strobe called $POC. Strobe $POC eventually generates $ZOR and $ORY, although not necessarily both at the same time. Resetting the B flip-flop and thereby sending the even operation code to the Operations Unit is accomplished when all the terms for an even OU instruction are true; specifically:

$$\begin{aligned} \text{B reset} = \$\text{POC}\cdot[(\overline{\text{ONR}}\cdot\text{EOC}\cdot\text{EOF}) && (1)\\ \cdot\text{TRMF}\cdot\text{EIF}\cdot\text{IC}\cdot(\text{Y}+\text{TSX}+\text{ROS})\cdot\overline{\text{TMCH}=1}] && (2)\end{aligned}$$

1) This equation represents the general case for resetting the B flip-flop and sending the even operation code for an operation unit instruction to the Operations Unit.

2) These terms represent a special case where a terminate for a "repeat" type instruction, occurs after the process to send the operation code for a fast operations unit type instruction to the Operations Unit, has already started. For this condition, it is too late to stop the generation of $POC and the B flip-flop from being reset and sending the even operation code; so B is inhibited from being reset. The B flip-flop is not changed because the Operations Unit cannot accept an operation code until housekeeping and cleanup functions for the "repeat" instruction takes place within the Operations Unit.

When all the terms for resetting the B flip-flop are present and $POC is generated, $ZOR is generated to strobe the even and odd operation code into the RE and RO buffers. Strobe $ORY is also generated by $POC, but occurs approximately 144 nanoseconds after $ZOR.

$$\$\text{ZOR} = \$\text{POC}\ (\text{EVEN}+\text{NOS}\cdot\text{ODD}) \qquad (3)$$

3) Indicating that $ZOR is generated for an even operations unit instruction odd operations unit instruction that was preceded by a nonoperations unit instruction. When $ZOR is generated, both the even and odd operation code is strobed into the RE and RO buffer.

9.3.7.2 - contd

Whenever \$ZOR is generated, it is delayed 54 or 90 nanoseconds and used to strobe ICTA' with the address of the instruction waiting in the RE-buffer (even instructions) or RO-buffer (odd instruction when preceded by a nonoperation unit instruction).

The conditions which set the B flip-flop are:

$$B\ set = \$\underline{POC\ [(\overline{ONR}\cdot OOC\cdot \overline{EOF})\qquad\qquad} \qquad (4)$$

$$\cdot\overline{TRMF}\cdot EIF\cdot IC\cdot (Y+TSX+ROS)\cdot TMCH=1] \qquad (5)$$

4) This equation represents the general case for setting the B flip-flop, when the odd operation code is going to be sent to the Operations Unit.

5) The groups of terms in this equation are identical to equation (2) with the exception of IC which represents the odd rather than the even instruction.

As before, when a \$POC is generated \$ORY is also generated: but \$ZOR will not necessarily be generated.

The conditions which set the B flip-flop are:

$$B = \overline{ONR}\cdot OOC\cdot \overline{EOF}$$

In this case, the B flip-flop is set, \$POC is generated, but no \$ZOR unless the even operation code had been a nonoperation unit (NOS true) instruction. However, \$ORY is generated approximately 100 nanoseconds after \$POC.

Control ofthe RE and RO switches to the ZOR bus is a function of the B flip-flop. However, the B flip-flop outputs may be inhibited by the RCU and RTD instructions.

$$¢\ RE/ZOR = \overline{B}\cdot ((\overline{RCU+RTD)\cdot GBO})$$

$$¢\ RO/ZOR = B\cdot ((\overline{RCU+RTD)\cdot GBO})$$

When (RCU+RTD)·GBO is true, the controls ¢ RE/ZOR and ¢ RO/ZOF, are inhibited and a return instruction (630_8) is forced on the ZOR bus, so the Operations Unit can load its registers and indicators.

9.3.7.3 Timing Diagram

Figure 9.3.7.3 is a timing diagram for the sequences that take place in loading the RE- and RO-buffers, and is representative of the normal case, that is, both RE and RO-buffers, and is representative of the normal case, that is, both RE and RO loaded at the same time.

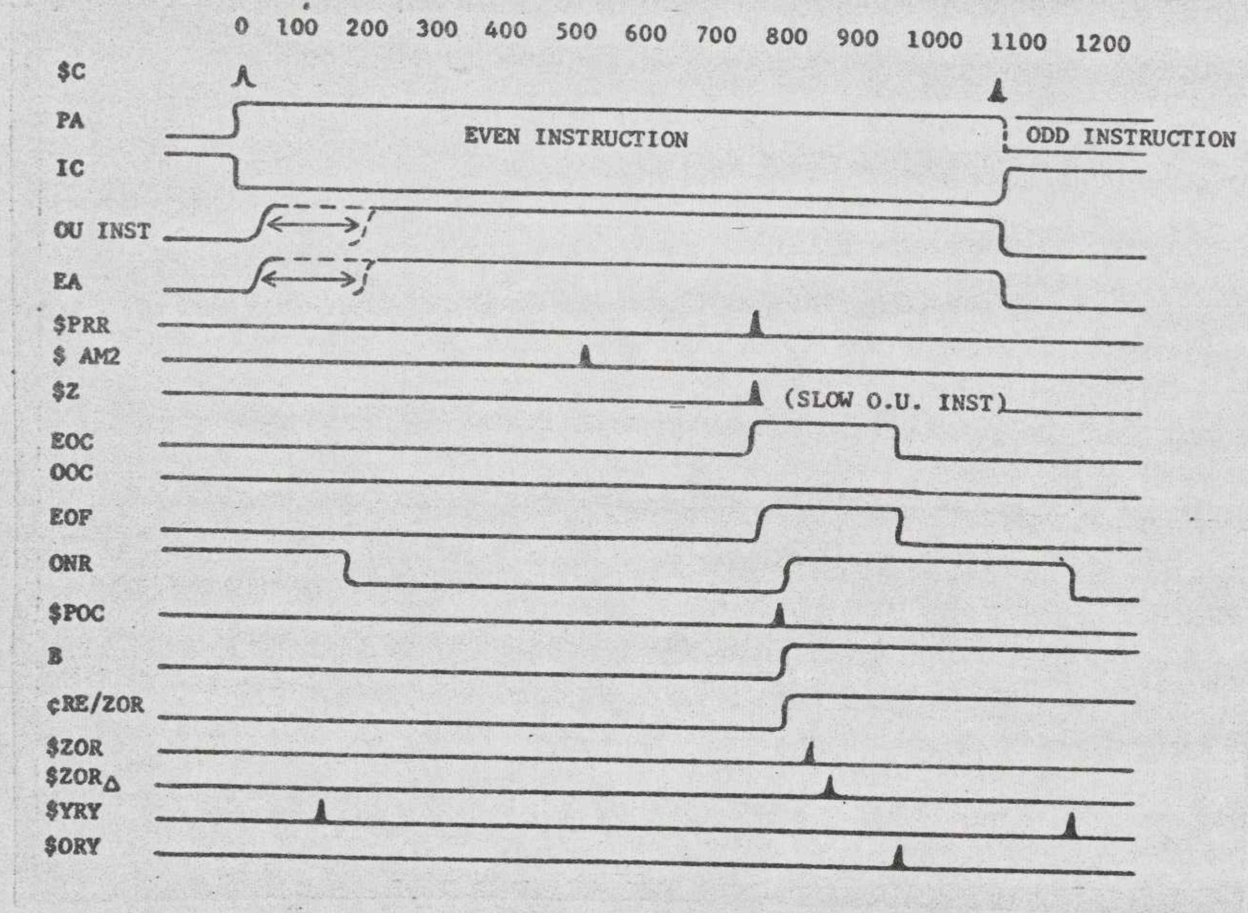
9.3.8 MFREE COUNTER AND MRD CONTROL (See 43D163387 sheet 11)

As mentioned previously, the Control Frame has the capability of being one, two or three instructions ahead of the Operations Unit. In addition, all operands sent to the Operations Unit are buffered in the M-register of the Operations Unit until they are moved into the working registers. However, during multiply operations, the M-registers is used as a working register. This means that the Control Frame must continually check the status of the M-register before attempting to send an operand to the Operations Unit. The Control Frame is provided with a MFREE counter to keep track of the number of operands sent to, or on the way to, the M-register of the Operations Unit. This register keeps track of up to three operands; one in the M-register of the Operations Unit, one in the Direct Buffer in the Base Frame, and one on the way from memory. The third operand will only be requested from memory if the Operations Unit is doing a "short" instruction and will be free when the operand arrives.

9.3.8.1 Count-Up Cycle

The count-up cycle input for each MFREE counter stage is basically the same for each stage as can be seen from the logic diagram.

Each stage is set as successive operations unit operands are requested from memory. Counting of the operands is as shown on the logic diagram and in Table 9.3.8.1


0
100
200
300
400
500
600
700
800
900
1000
1100
1200
$C
PA
IC
EVEN INSTRUCTION
ODD INSTRUCTION
OU INST
EA
$PRR
$ AM2
$Z
(SLOW O.U. INST)
EOC
OOC
EOF
ONR
$POC
B
¢RE/ZOR
$ZOR
ZOR_\Delta$
$YRY
$ORY

NOTE:

1) It is assumed that the Y- and C-registers are loaded with slow Operations Unit instructions and the Y-register operand address will be prepared first.

FIG. 9.3.7.3 RE, R0 Buffer Loading, Timing Diagram

Table 9.3.8.1 MFREE Counter States

Count	$MFREE_2$	$MFREE_1$	$MFREE_0$
0	0	0	0
1	0	0	1
2	0	1	1
3	1	1	1

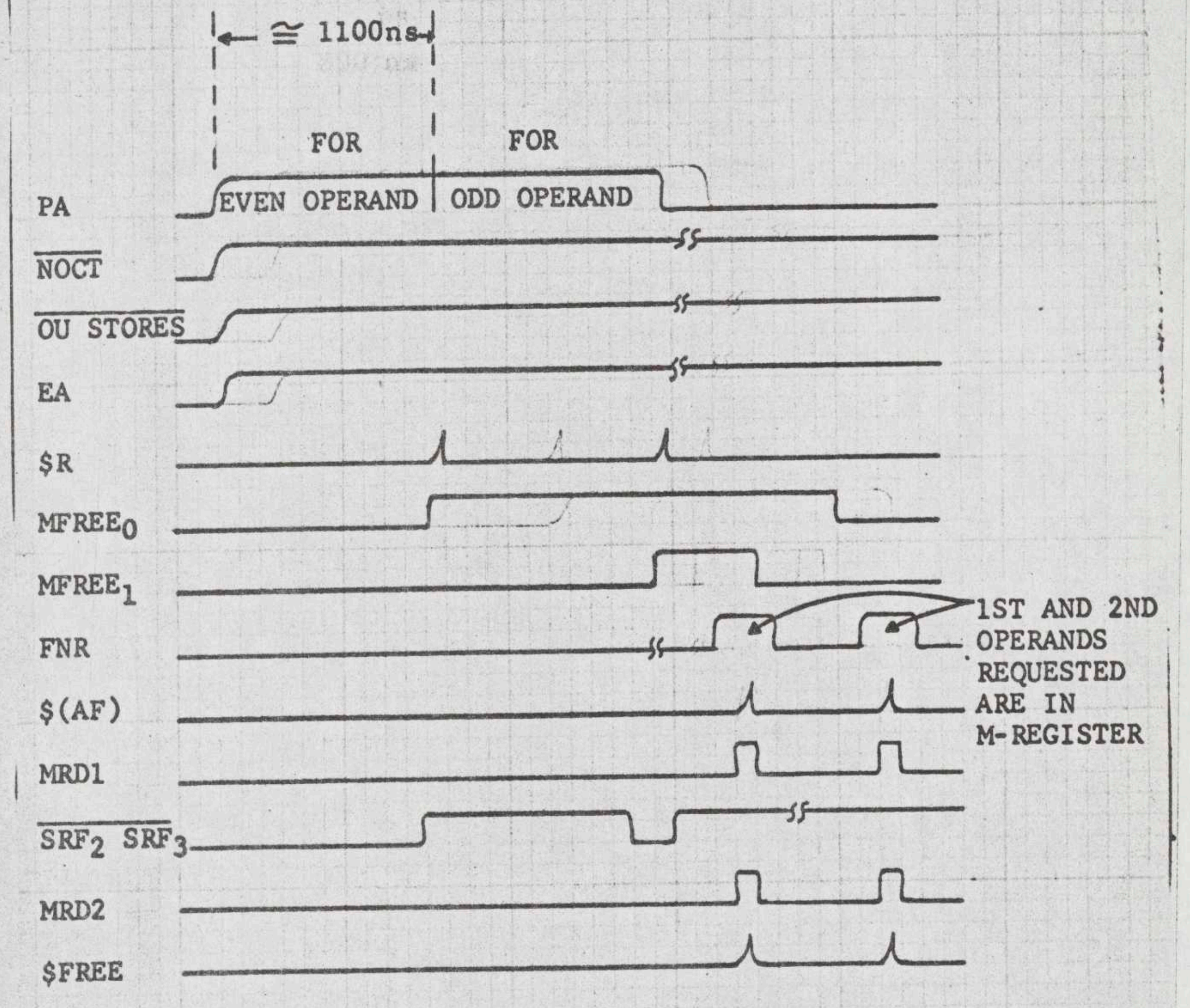
9.3.8.2 Count-Down Cycle

The count-down cycle is somewhat different than the count-up cycle due to the asynchronous nature of the Control Frame and the Operations Unit, and the possibility of trying to count down and up at the same time. Two flip-flops, MRD1 and MRD2, are used to generate a \$FREE which resets or counts down each stage of the MFREE counter, when the M-register signals that it is free. Counting down begins with stage $MFREE_2$ and will end with stage $MFREE_0$, if no new operands are fetched for the Operations Unit before the M-register is free.

The MRD1 flip-flop sets with strobe \$(AF)·(FNR). Strobe \$(AF) is general strobe generated within the Operations Unit; $\overline{FNR}$ indicates that the Operations Unit cannot accept a direct operand. Generally, FNR is true. When MRD1 sets, it will set MRD2 and generate \$FREE to count down the MFREE counter, provided the term $\overline{SRF_2 \cdot SRF_3}$ is true. This term indicates that a \$R has been issued for an operand and MFREE has been counted up. If $SRF_2 \cdot SRF_3$ is true, the MRD2 stage will not be set until $\overline{SRF_2 \cdot SRF_3}$, to allow MFREE to be counted up before being counted down, thus preventing a race condition. Strobe \$FREE resets MRD1 which resets MRD2, and all circuits are ready to respond to the next \$(AF)·(FNR) combination indicating the M-register is free.

9.3.8.3 Timing Diagram

Figure 9.3.8.3 is a timing diagram of the MFREE count-up and count-down cycles.


≅ 1100ns
FOR
FOR
EVEN OPERAND
ODD OPERAND
PA
NOCT
OU STORES
EA
$R
MFREE$_0$
MFREE$_1$
FNR
$(AF)
MRD1
SRF$_2$ SRF$_3$
MRD2
$FREE
1ST AND 2ND OPERANDS REQUESTED ARE IN M-REGISTER

NOTES:

1. Two operands are requested from memory before the first operand is used by the Operations Unit.
2. Counter shows count of two, then, is decremented.
3. Relationships are not to scale.

FIG. 9.3.8.3 MFREE Counter, Timing Diagram

9.3.9 TMCH COUNTER (See 43D163387, sheets 43, 44)

The Control Frame contains a three-stage counter to keep track of the number of operands currently in, or on the way to the Operations Unit as distinguished from the MFREE counter which keeps track of the operands sent to the M-register. This counter is deisgnated TMCH, and is counted up every time an operand is sent to the Operations Unit, as signified by receipt of $R (and certain other conditions) generated for an interrupt. Counting down of TMCH is done by a $DOWN which is a function of conditions which indicate that the Operations Unit has finished or will not complete an operation.

9.3.9.1 Count-Up Cycle

The three stages of the TMCH counter are set by the following equations:

set conditions

$$\mathrm{TMCH}_0 = (\mathrm{EA}\cdot\overline{\mathrm{TMCH}_1\Delta}\cdot\overline{\mathrm{NOCT}})\cdot\$\mathrm{R}$$

$$\mathrm{TMCH}_1 = (\mathrm{EA}\cdot\mathrm{TMCH}_0\Delta\cdot\overline{\mathrm{NOCT}})\cdot\$\mathrm{R}$$

$$\mathrm{TMCH}_2 = [(\mathrm{EA}\cdot\mathrm{TMCH}_1\Delta\cdot\overline{\mathrm{NOCT}})+(\mathrm{EA}\cdot\mathrm{TMCH} = 1\cdot\mathrm{NOP})]\cdot\$\mathrm{R}$$

This shows that the TMCH counter will be counted up one stage at a time with each interrupt for an operations unit operand. There is a special case where TMCH will indicate a count of 1+NOP (No Operation Performed). Table 9.3.9.1 shows the count in the different stages for all conditions.

Table 9.3.9.1 TMCH Counter States

Count	$TMCH_2$	$TMCH_1$	$TMCH_0$
0	0	0	0
1	0	0	1
2	0	1	1
3	1	1	1
1+NOP	1	0	1

The TMCH counter will only be allowed to go to three, in the case of short operations unit instructions.

9.3.9.2 Count-Down Cycle

The count-down cycle is quite different from the count-up cycle because of the asynchronous nature of the Operations Unit and the Control Frame, and the possibility of strobing TMCH up and down at the same time. Each stage is counted down (reset) beginning with $TMCH_2$, or when a "repeat" instruction is terminated as indicated by level DR.

reset conditions

$TMCH_2 = \$DOWN+DR$

$TMCH_1 = \$DOWN\cdot\overline{TMCH_2\Delta}+DR$

$TMCH_0 = \$DOWN\cdot\overline{TMCH_1\Delta}+DR$

Strobe $DOWN and the conditions which generate is are defined as follows:

$$\$DOWN = \Delta C\cdot(TRMF\cdot\overline{WN}) \quad (1)$$

$$\Delta C = \$PDN \quad (2)$$

$$\$PDN = LST_2\cdot(SEP\cdot\overline{DVCK}\cdot FOLF\cdot LST_1\Delta) \quad (3)$$

$$LST_2 = \overline{SRF_2\cdot SRF_3\cdot SFF}\cdot LST_1 \quad (4)$$

$$LST_1 = (\$LAST+\$FOLF+\$DVCK)\cdot\overline{WF} \quad (5)$$

9.3.9.2 - contd

Analyzing the equations from number (5) and working toward the generation of $DOWN:

5) $LAST, $FOLF and $DVCK indicate that the Operations Unit is finished with an operand, or there is a overflow or divide check for an arithmetic operation which also terminates operations unit action. Any one of these strobes unconditionally sets LST_1, unless a fault is about to be processed.

4) LST_2 is set whenever SRF_2 and SRF_3 do not occur together and no fault has been detected. This effectively inhibits any action to generate $DOWN beyond this point.

$$SRF_2 \text{ set} = (EA\cdot\overline{T})\cdot\$PRR$$
$$\text{reset} = \$R_\Delta$$
$$SRF_3 \text{ set} = \$IF+\$SSRF$$
$$\text{reset} = \$R_\Delta+\$RSRF$$

These equations show that LST_2 will be set whenever $R is not about to be issued for an interrupt. This $R would count up the TMCH counter, at possibly the same time that it was being counted down. That is, SRF_2 "sets" on the interface term coming true for a operations unit operand, SRF_3 is set on $IF and reset by $RSRF if the Interface Frame must perform an appending operation. Strobe $SSRF from the Interface Frame then sets SRF_3 again, inhibiting the setting of LST_2 and stopping $DOWN. Resetting of both SRF_2 and SRF_3 will occur at $\$R_\Delta$ time; approximately 100 nanoseconds after the $R for the current interrupt has counted up the TMCH counter. IF LST_1 is set when SRF_2 and SRF_3 are reset, LST_2 is set. Summarizing: <ST_2 is set only if there is no chance of $R for the interrupt, and $DOWN resulting from a $LAST, strobing the TMCH counter at the same time.

9.3.9.2 - contd

3) Strobe $PDN (pre-down) is generated as a result of LST_2, LST_1 and not having any of the other conditions:

SEP -- Instruction with parity error was not completed.

DVCK -- Divide check on arithmetic operation of last operations unit instruction.

FOLF -- Overflow on the arithmetic operation last executed by Operation Unit.

2) The level ΔC, initiated by $PDN and terminated by ΔC delayed through additional delay inverters to its own reset input, is to allow enough time for portions of the Z-function logic to be reset before TMCH is strobed down.

1) Finally, if there is a TRMF·$\overline{WN}$ condition, no $DOWN is generated, unless TRMF·$\overline{WN}$ comes true before ΔC resets. When the terminate condition is serviced for a "repeat" type instruction, then a level called DR is generated which resets the TMCH counter to a zero state.

9.3.10 STORE COMPARE (EX) FUNCTION (See 43D163387 sheet 37)

The store-compare function (EX-function) indicates that the current instruction being processed will store data into one of the memory locations from which the remaining instructions contained in the Control Frame are located. If such is the case, the instructions following the one currently being prepared by the Control Frame must be refetched. Whether one, or two instructions must be refetched depends on the results of the comparison between the current instruction being prepared, and the remaining instructions. The comparison is made by comparing the operand address of the instruction in the Y-register to the memory location of the instructions in the C-register (EX_1 level) and the I-register (EX_2 and EX_3 levels). Similarly, when preparing the instruction in the C-register, the operand address is compared

9.3.10 - contd

to the memory location of the instructions in the I-register (generation of EX_2 and EX_3 levels only).

The comparisons are made in the following manner and will cause the following events to occur:

EX_1 level:
1) $ICTC_{0-16}$ compares with BS_{0-16}.
2) BS_{17} = "1" indicating the address is for the C-register or a STAQ instruction is decoded which effects the C-register.
3) The PBR contents equals the TBR contents indicating that the addresses of the instructions compared are in the same segment.
4) The Control Frame is preparing a store or read-alter-rewrite type instruction.
5) The Y-register did not contain an execute double instruction. This is to prevent a store-compare level when an XED instruction is "brute forced" into the Y- and C-registers because of a fault.

When the conditions in 1 through 5 above are satisfied, the EX_1 level is generated and causes:

6) An XEC instruction with an R modification (designating the ICTC register contents to repull the instruction for the C-register) to be "brute forced" into the C-register.
7) Inhibits the strobing of the Y-register for the refetched instruction coming into C-register via the V-switch, ZDI bus, and VTB switch.

9.3.10 - contd

EX_2, EX_3 levels.

8) $ICTB_{0-16}$ compares with BS_{0-16}.

9) The PBR contents equals the TBR contents indicating that the addresses of the instructions compared are in the same segment.

10) The Control Frame is preparing a store or read-alter-rewrite instruction.

When the conditions in 8 through 10 are satisfied and the EX_2, EX_3 levels are generated, the following occurs:

11) The EXI flip-flop is set to remember that a comparison was made on the I-register.

12) If the comparison was made when preparing the instruction in the Y-register, the instruction in the C-register is executed first, then the I-register is repulled using the ICTB address that is sent to the BS-adder. As the instructions are brought in they are routed around the I-register to the Y- and C-registers. When EX_2 and EX_3 is generated, the ICTB address is not updated in the normal manner (adding +1 to both the YS- and BS-adders) until the EX_2 and EX_3 levels are removed and the EXI flip-flop is reset. The EXI flip-flop was used to generate EX_2 and EX_3. In cases where the instruction in the C-register is being compared against the I-register, and EX_2 and EX_3 levels are generated, no waiting is necessary and the I-register instructions are repulled as previously described.

9.3.10 - contd

In those cases where the I-register is normally refetched (transfers, PI cycles or faults) the EXI flip-flop is reset, since the ICTB address is either different or invalid.

9.3.11 Z FUNCTION (See 43D160212 sheet 5, and 43D163387 sheets 19,20

The purpose of the Z-function is to hold up any address modification for an instruction being prepared by the Control Unit, when the Operations Unit is changing a register that is required for the address modification. Only R, and RI type address modification in the instruction word and IR modification in the indirect word is considered, since only these modifiers can designate a register that may be required by the Operations Unit to execute an instruction; specifically, the X0-X7, A- and Q-registers. Generation of the Z-function is accomplished by comparing the tag field of the instruction or indirect word currently being prepared, that is on the $\mathrm{DW}\frac{00}{08}$ 1B (ZI) bus, with the coded operation code of the previous one or two instructions that had been sent to the Operations Unit. These instructions are saved in the Z1 and Z2 registers (see the Z-function circuitry on the upper righthand corner of 43D160212 sheet 5). The specific code of the tag field on the ZT bus and coded operation code field that is saved in the Z1 and Z2 registers is shown in Table 9.3.11. Note in Table 9.3.11, that the RI and IR address modification types IC, DU, DL and N are omitted since these modifications do not use the X-, A- or Q-register.

Table 9.3.11 Comparison of Codes on ZT Bus with Codes in Z1, Z2 Registers

	Tag				Code in Z1, Z2 (Note 1)				
Tag Reg. Designates	ZT_0	ZT_1	ZT_2	ZT_3	ZI_0	ZI_1	ZI_2	ZI_3	Reg. Designated by Code in Z1
X0	1	0	0	0	1	0	0	0	X0
X1	1	0	0	1	1	0	0	1	X1
X2	1	0	1	0	1	0	1	0	X2
X3	1	0	1	1	1	0	1	1	X3
X4	1	1	0	0	1	1	0	0	X4
X5	1	1	0	1	1	1	0	1	X5
X6	1	1	1	0	1	1	1	0	X6
X7	1	1	1	1	1	1	1	1	X7
AU	0	0	0	1	0	0	0	1	A
AL	0	1	0	1	0	0	1	1	AQ
QU	0	0	1	0	0	0	1	0	Q
QL	0	1	1	0	0	0	1	1	AQ

NOTE:

1) Coding of Z1 is identical to Z2.
2) Register designators for ICTC, DU, DL and N are ignored.

9.3.11 - contd

Generation of the Z-function (if one is to be generated) is done in the following manner: (See 43D160212 sheet 5, and 43D163387 sheets 19,20)

1) An instruction is being prepared and the resultant address preparation cycle causes the operation code to be placed on the ZI bus (ZI_{0-8}).

2) When the preparation cycle is completed and a $R is generated, the Z1-register is strobed with the appropriate bits of the ZI bus that may designate one of the X0-7 or A- or Q-registers. The bit positions being selected are the same for all instructions and represent all cases where the X0-7, A- and Q-registers are designed by the operation code.

3) Address preparation is now started for the next instruction. At this time the tag portion of the ZI bus is switched to the Z comparator via the ZI_{14-17} (ZI/ZT) switch.

4) If Z1 compares with the ZT bus then a BZFN1F (Z-function) level is generated. If no comparison no level.

5) The Z-function level in conjunction with $∧AM1, sets a DZ flip-flop in the $R, $C generation circuits which inhibits the generation of $IF, and the resultant strobes to start the Interface Frame. When the Interface Frame is not started no $IFF is received in the Control Frame to set the IFR flip-flop in the $R, $C generation circuits and any further strobes are inhibited. (See paragraph 9.3.1 for this discussion in detail.)

6) No further address preparation takes place until the Operations Unit sends a $LAST. This strobe causes the TMCH counter circuits to generate a $DOWN that eventually resets the DZ flip-flop and caused strobe $IF to be generated, starting the strobe chain again. Resetting the ZI register takes place as a function of level ΔC (generated in the process of producing a $DOWN) and the TMCH counter equal 1, or 1 + NOP state.

9.3.11 - contd

The foregoing steps (1 through 6) presuppose that the Operations Unit was not busy and that there was a comparison on the Z1-register. If there was no comparison on the Z1-register, the following takes place:

7) The Z1-register is moved into the Z2-register concurrent with strobe $PRC (a strobe that precedes $R).

8) The Z1-register is strobed by $R and then will contain the decoded operation code field of the instruction that is currently being processed and present on the ZI bus.

9) When address preparation is started for the third instruction (two instructions sent to the Operations Unit, and two codes in the Z1-, Z2-registers) the tag field of this instruction is compared to the Z1- and Z2-registers and possible generation of the Z-function.

10) When the Operations Unit produces a $LAST resulting in a $DOWN, the Z2-register is reset, since the Operations Unit is finished with the instruction that is represented by the code in the Z2-register.

11) The sequence of steps continue until the Operations Unit is caught up to the Control Frame (as far as instructions that effect the X0-7, A- and Q-registers) or the tag fields of new instructions do not specify the X0-7, A- and Q-registers.

For instructions that involve RI or IR address modification, the new tag(s) brought in by the indirect word(s) are compared with the Z1- and Z2-registers, but the operation codes in the Z1- and Z2-registers are not changed until a new instruction is being processed. There is a special case when the Z-function is generated for "repeat" type instructions. Whenever one of the repeated instructions designates any index register, the Z-function is generated until the Operations Unit is finished with the current instruction execution.

9.3.11 - contd

When an index register is specified by the operation code and tag field of one of the repeated instructions, both the index register designated by the instruction and the index register designated by tag field of the instruction must be changed. To do this, the Operations Unit executes the index instruction; then the delta portion of the "repeat" instruction is added to the index register used for address modification of the repeated instruction(s) (see the RPT, RPD and RPL instruction descriptions for address modification of the repeated instructions.) This means that for "repeat" type instructions with address modification, the Control Unit never gets ahead of the Operations Unit.

9.3.12 MAINTENANCE/CONTROL PANEL FUNCTIONS

A re-examination of the Maintenance/Control Panel covered in Section 3.0 of Part I of this EPS shows that there is a considerable array of indicators, switches and pushbuttons. Only those switches that are directly involved with the Control Frame circuits will be discussed. The indicator lamps for the various registers and status flags are not involved with any circuit actions within the Control Frame, since the voltages used to control the indicator lamps are tapped from the circuits involved without effecting any operation. The only circuit actions that take place, allow different registers to be displayed on the same set of indicator lamps. This is done by manual switching from the Maintenance/Control Panel proper (see 43D139956). The following switches and pushbuttons on the Maintenance/Control Panel effect the Control Frame operation: REPEAT EXECUTE; STOP; OU/CU OVERLAP OFF; CF TEST CLOCK; EXECUTE SWITCHES; EXECUTE (pushbutton); STEP (pushbutton); TEST ENABLE; and STEP ENABLE rotary switch positions: OFF, PROG, CU and MEM.

9.3.12.1 Switch Functions (Refer to 43D163387 sheets 39,40,41 for details)

TEST ENABLE - when this switch is activated the switch functions on the Maintenance/Control Panel are enabled.

9.3.12.1 - contd

This enable level is used in various places throughout the Control Frame circuitry as can be noted from the logic diagrams.

CF TEST CLOCK - When this switch is activated, the generation of strobes in the \$C, \$R generating circuits is speeded up by approximately 18 nanoseconds for marginal timing checks.

OU/CU OVERLAP OFF - When this switch is activated, the Control Frame will not proceed in advance of the Operations Unit. This is accomplished by inhibiting the interface term for strobe generation until the TMCH counter equals zero.

STOP - This is a momentary ON switch. When depressed, the STOP flip-flop is set. When STOP is set, a fault sequence is initiated by the processor fault logic in the Base Frame, the normal "snapshot" sequence in preparation for fault servicing takes place, and the Processor stops in a WF cycle. When the STOP switch is released a DIS instruction (616_8) is forced onto the ZDI bus via the normal input data switch (V switch). The STOP flip-flop is reset by \$C·WF and the DIS instruction is "brute forced" into the Y- and C-registers. A PA cycle is entered for the DIS instruction, then terminated by the next \$C produced from the \$C, \$R logic. The result is the Processor initialized and waiting for an interrupt.

EXECUTE (pushbutton) - Depressing the EXECUTE pushbutton initializes the Processor by setting a pre-stop (PST) flip-flop that forces the STOP flip-flop to set thereby causing the same actions as depressing the STOP switch. When PST resets itself, approximately 180 nanoseconds later, a DIS instruction is forced on the ZDI bus and the Processor is initialized. Releasing the EXECUTE pushbutton causes the EXF flip-flop to be set, causing the processor fault logic to initiate an execute fault. The normal "snapshot" and fault sequence is entered, resulting in a "brute-forced" XED instruction with a fault address corresponding to the execute fault-vector pair in memory, forced into

---

9.3.12.1 - contd

the Y- and C-registers. Any action once the fault-vector pair is pulled from memory depends on the instructions contained in the fault-vector.

EXECUTE SWITCHES - When the EXECUTE SWITCHES switch is activated, the contents of the INSTRUCTION switches on the Maintenance/Control Panel are forced into the Y- and C-registers instead of the instructions contained in the fault-vector pair in memory corresponding to the execute fault.

REPEAT EXECUTE - The REPEAT EXECUTE switch works in conjunction with an oscilloscope gate input at the processor SYNC PANEL located on the physical cabinet frame behind the Maintenance Panel. Application of an oscilloscope gate as a trigger source, and with the REPEAT EXECUTE, and EXECUTE SWITCHES, switches activated, causes the Processor to be initialized, do an execute fault from the INSTRUCTION switches and keep repeating the sequence.

STEP pushbutton - Depressing the STEP pushbutton sets the SPR flip-flop. When SPR is set it allows $PRII which generates $R (providing all other strobe generation conditions are met) and starts the Processor. SPR is normally set when the Processor is running and is only reset with $PRR and the conditions generated by the STEP ENABLE switch.

STEP ENABLE switch:

PROG position - strobe $PRII's are stopped since SPR is reset in the first PA cycle entered by the Processor preventing any more $R's from beinggenerated by the $R, $C generation circuits.

CU and MEM positions - this position stops $R's in the same manner and the PROG position except that SPR is reset during any prepare cycle except PR.

The net effect is that for the PROG position, the Processor can step from $C·EA to $C·EA; in CU position from $C to $C; and in MEM position from $INT to $INT.

9.4 BASIC OPERATING SEQUENCES

To be supplied later

10.0 APPENDING UNIT DETAILED OPERATION

The Appending Unit contains the hardware registers and control logic to handle all functions concerned with the selection of address pointers to segments and pages of segments in memory, and the actual communication between the Processor and Memory. To perform the necessary functions requires two frames of logic: the Interface Frame and the Base Frame. The Interface Frame is concerned with the final address preparation for data to and from memory, and the Base Frame is concerned with storing the address pointers in one of its various registers and supplying this information when requested by the Control Unit.

Additional functions performed by each frame is explained under the applicable description of the particular frame. One note of information, however; the majority of the Processor fault logic is contained in the Base Frame due to space limitations in the Control Unit. Pertinent information is discussed following the Base Frame Detailed Operation.

10.1 INTERFACE FRAME DETAILED OPERATION

The Interface Frame, as its name implies, is the link or interface between the Processor and the Memory Controller(s). The processor must be capable of communicating with from one to eight Memory Controllers with various size core memories. It must be able to select the proper Memory Controller and to respond to signals from any of the Memory Controllers.

The Interface Frame consists of address preparation logic, data out logic, and memory selection and response logic. It handles all communication with the Memory Controllers except receipt of data on the Data In bus, which is handled by the Base Frame. It performs the final address preparation in one of three modes, and makes all fault checks having to do with that address. It develops all control signals for, and sends them to, the proper Memory Controller. It responds to all control signals from the Memory Controller.

10.1.1 FUNCTION

The Interface Frame consists of three major groups: Address Preparation, Data Output, and Memory Control.

The address preparation logic is constructed around two registers, an adder, and a comparitor. The Descriptor Register (DES_{0-35}) receives all address appending words from core memory (for example, PTW). The format of this register is the same as for the word that it holds. This register and its associated logic receives a SDW, PTW, etc. from the core memory via the ZDI lines in the Base Frame. The data is held for use by the adder and is sent, for storage, to the Associative Memory. The Descriptor Generator generates the descriptor field for PTWs and checks for all faults except Out of Bounds. The Address Register (ADR_{0-23}) receives all address of words to be retrieved for core memory. This address is used by the memory selection logic and is sent to all Memory Controller ports. The RS-adder (RS_{0-17}) is used in the preparation of addresses. It accepts addresses or portions of addresses from the DES Register, the Base Frame, the Control Frame, the Associative Memory, and the Maintenance Panel, and adds or concatenates the pieces to form the final address, which is stored in the ADR Register. The Boundary Comparitor (BC_{0-11}) checks the address against the boundary field except in Absolute mode, where it checks the address against the upper limit of physical core memory. This logic is able to operate in any one of three modes as determined by the start pulse received from the Control Frame.

Absolute Mode (\$SAB)--In this mode the address from the Control Frame is the final address and it is merely passed through the RS-adder and stored in ADR_{0-23}. The address is checked by BC_{0-11} to see if it is within the physical limits of memory. If it is not, an OOB faults occurs.

10.1.1 - contd

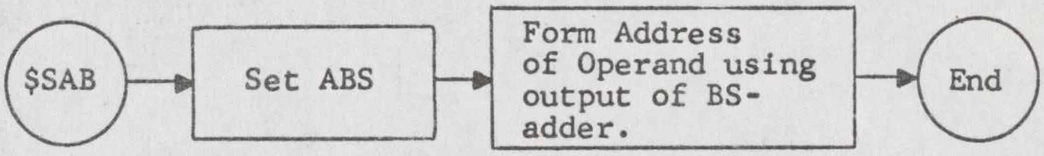


Fig. 10.1.1 Absolute Mode Addressing, Flow Chart

Append Without Associative Memory ($SAP)--In this mode the address from the Control Frame must go through the appending process. During the appending the Associative Memory does not operate and all information is retrieved from core memory. All appending words are checked for faults and the final appending word is checked for access violation. The final address is checked against the boundary field to see if it is within the specified boundary. If it is not, an OOB fault occurs. See Fig. 10.1.1a.

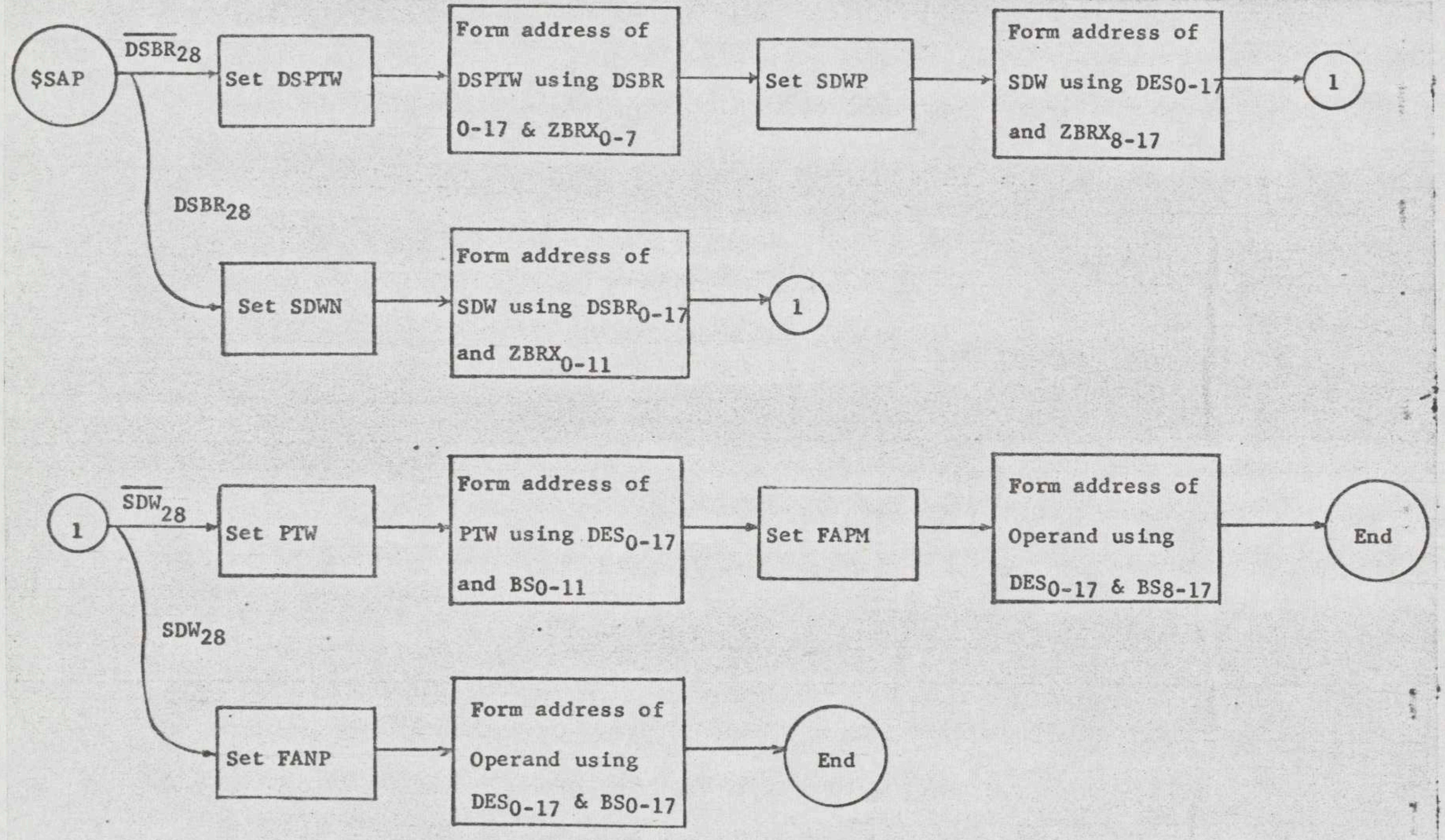

$SAP
$\overline{DSBR_{28}}$
Set DSPTW
Form address of DSPTW using DSBR 0-17 & $ZBRX_{0-7}$
Set SDWP
Form address of SDW using DES_{0-17} and $ZBRX_{8-17}$
1
$DSBR_{28}$
Set SDWN
Form address of SDW using $DSBR_{0-17}$ and $ZBRX_{0-11}$
1
1
$\overline{SDW_{28}}$
Set PTW
Form address of PTW using DES_{0-17} and BS_{0-11}
Set FAPM
Form address of Operand using DES_{0-17} & BS_{8-17}
End
SDW_{28}
Set FANP
Form address of Operand using DES_{0-17} & BS_{0-17}
End

Fig. 10.1.1a Append Without Associative Memory Flow Chart

10.1.1 - contd

Append with Associative Memory ($AAM)--In this mode the address from the Control Frame must go through the appending process. The Associative Memory will be searched, and if the required information (PTW or SDW) is located it will be used in the address preparation. If the information is not there it must be retrieved from core memory. Only the appending words retrieved from core memory are checked for faults. All final appending words are checked for access violations. See Fig. 10.1.1b.

There is also a special case dealing with the written bit. Whenever a page is altered by a store or RAR operation the written bit of its PTW must be set to a "1." If the PTW was retrieved from core memory, this presents no problem but if it was obtained from the Associative Memory, a special sequence must be gone through. Even though the PTW is in the Associative Memory, a special sequence must be gone through. Even though the PTW is in the Associative Memory its address in core memory must be developed in order to set its written bit.

The final address is checked against the boundary field to see if it is within the specified boundary. If it is not, an OOB fault occurs.

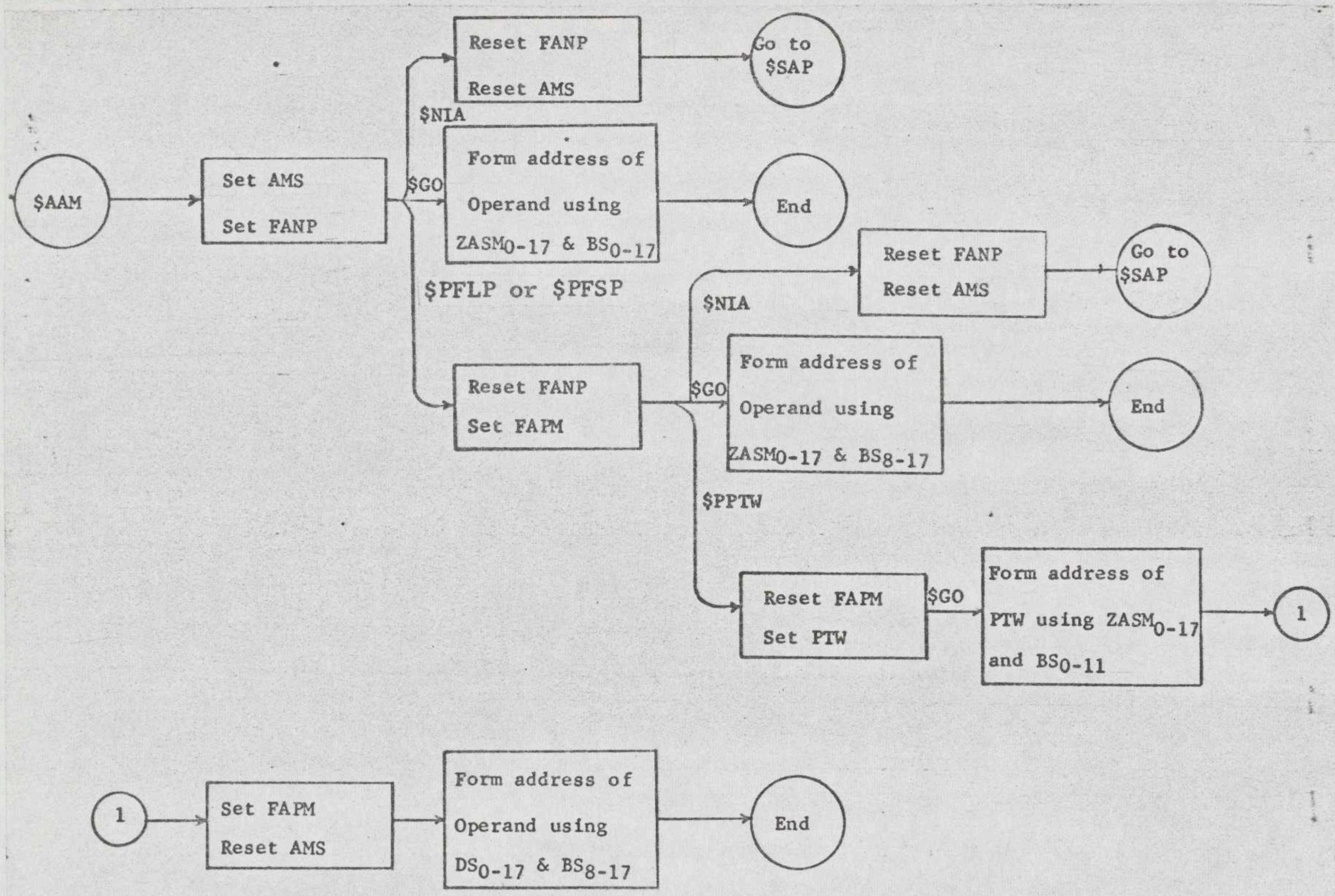

$AAM
Set AMS
Set FANP
$NIA
Reset FANP
Reset AMS
Go to $SAP
$GO
Form address of Operand using ZASM0-17 & BS0-17
End
$PFLP or $PFSP
Reset FANP
Set FAPM
$NIA
Reset FANP
Reset AMS
Go to $SAP
$GO
Form address of Operand using ZASM0-17 & BS8-17
End
$PPTW
Reset FAPM
Set PTW
$GO
Form address of PTW using ZASM0-17 and BS0-11
1
1
Set FAPM
Reset AMS
Form address of Operand using DS0-17 & BS8-17
End

Fig. 10.1.1b Append With Associative Memory, Flow Chart

10.1.1 - contd

The data out logic is constructed around a 72-bit register (DOR_{0-71}). The input to this register comes from the ZDO lines of the Base, Control, and Register Frames, from the Read Out switch of the Associative Memory, and from the RS Adder. Any information that is to be stored in core memory will be placed in this register (except written bits). It is also used during the "snapshot" portion of a fault to retain certain information until it can be stored by a SCU instruction. The output of this register goes to a switch that selects either the upper half (DOR_{0-35}) or the lower half (DOR_{36-71}) and presents it to all memory ports.

The memory control logic is constructed around various control flip-flops, registers, and strobe generators. The DIR1 and DIR2 flip-flops are used to determine which one of the eight memory ports will be used. The registers are:

1) Command (CMD_{0-3}) -- The specific action to be taken by the Memory Controller is determined by this register.

2) Zone ($ZONE_{0-7}$) -- The particular portion of a word to be altered is determined by this register.

3) Illegal Action (IAL_{0-2}) -- If an illegal action has occurred in the Memory Controller this register will be set with a code indicating what occurred.

10.1.1 - contd

This portion of the logic must accept an operation code from the Control Frame and an address from its own address preparation logic, determine what is needed with the core memory (for example, appending word, read or store operand, instruction) and set up its Command Register to tell the Memory Controller what to do. It must also determine if a 6- or 9-bit character operation is in progress, and if so, set up the Zone Register accordingly. The IF must issue requests and other control strobes, and must recognize illegal actions and store the corresponding code.

10.1.2 INTERFACE

The Interface Frame has external interface with the Memory Controller(s) and internal interface with the other five frames.

10.1.2.1 External

The Interface Frame has identical connections to eight ports. It is connected to two of the three 50 pin connectors, which make up a port. Each port may or may not be connected to a Memory Controller. Open connections will have no affect on the associated logic. The following lines are used to send information to the Memory Controller(s).

1) Data Lines (36) -- These lines are used to transfer data from the Data Out Register.

2) Address Lines (24) -- The address that is prepared by the Interface Frame and stored in the Address Register is placed on these lines to specify the location in core storage that is to be involved in a read/write operation.

3) Zone Lines (8) -- These lines are used to select the 6- or 9-bit characters to be involved in any Clear-Write command or the rewrite portion of a Read-Alter-Rewrite command. A "1" on a Zone Line indicates that the corresponding character is to be altered. A "0" on a Zone Line indicates that the corresponding character is to remain unchanged. Table 10.1.2.1a correlates the Zone Line with the character and bit positions it controls.

10.1.2.1 - contd

Table 10.1.2.1a Zone Lines vs. Character and Bit Positions

Zone Line	Character	Bit Positions
0	0	0 - 5
1	1U	6 - 8
2	1L	9 - 11
3	2	12 - 17
4	3	18 - 23
5	4U	24 - 26
6	4L	27 - 29
7	5	30 - 35

4) Interrupt Line -- The Interface Frame selects the Memory Controller to be utilized and then sends out a pulse on this line to that Memory Controller. This pulse, called strobe interrupt, requests access to the memory.

5) Double Precision/Rewrite Line -- Double Precision Store and Read-Alter-Rewrite operations require two memory cycles. When the Interface Frame determines that the second 36-bit word of a Double Precision Store or the altered word of a RAR is on the data lines it places a pulse on this line signifying that the memory can start its second cycle.

10.1.2.1 - contd

6) Protect Line -- Certain operations involving memory can only be done if the Processor is executing a Master Procedure (including Absolute mode). The Interface Frame places a "1" on this line when the Control Frame is in Master mode.

7) Command Lines -- The Interface Frame receives the operation code from the Control Frame, determines the type of memory operation required and places that command code on these lines. There are ten possible commands as shown in Table 10.1.2.1b.

10.1.2.1 - contd

Table 10.1.2.1b Memory Commands

Mnemonic	Command	Code			
		A	B	C	D
RRS,SP	Read-Restore, Single Precision	0	0	0	0
RRS,DP	Read-Restore, Double Precision	0	0	0	1
CWR,SP	Clear-Write, Single Precision	0	1	0	0
CWR,DP	Clear-Write, Double Precision	0	1	0	1
RAR	Read-Alter-Rewrite	1	0	0	0
RMSK	Read Execute and Channel Interrupt Mask Registers	0	0	1	0
SMSK	Set Execute and Channel Interrupt Mask Registers	0	1	1	0
SXC	Set Execute Interrupt Cells	1	1	1	0
CON	Connect	1	1	0	0
XEC	Execute	1	1	0	1

The following lines are used to receive information from a Memory Controller:

1) Cycle Started Delayed Line -- A pulse (strobe PIN) on this line indicates that the Memory has granted access to the Processor, and that

---

10.1.2.1- contd

it has received the control information on the Command, Address, and Zone Lines. The Interface Frame need not retain the information on these lines past this pulse. On Double Precision or Read-Alter-Rewrite operations two pulses will be received by the Processor. The second pulse indicates that the control information for the second part of the operation has been received.

2) Data Available/Stored Line -- A pulse on this line has different meanings depending upon the operation being performed. During any read operation or the first half of a Read-Alter-Rewrite operation a pulse on this line indicates that the Memory has placed the required data on the Data Lines to the Processor and that it will remain there for 200 nanoseconds. During any store operation or the second half of a Read-Alter-Rewrite operation a pulse on this line indicates that the required data on the Data Lines to Memory has been received and that the Processor can now change these lines. During any Double Precision operation two pulses will be received one for each half of the double precision word.

3) Strobe Connect Line -- A pulse on this line indicates that the Memory Controller has received a connect command to this port from the Control Processor.

4) Illegal Action Strobe Line -- A pulse on this line indicates that the Memory has correctly completed its operation or that an illegal action has taken place. The code on the Illegal Action Lines tells which is which (see next paragraph). This pulse is used to sample the Illegal Action Lines.

10.1.2.1 - contd

5) Illegal Action Lines (3) -- These lines indicate the specific reason that a Illegal Action Strobe was issued. The code on these lines is valid only at the time that this strobe occurs. The codes and their meanings are:

A B C		Priority
1 0 0	Not Control Processor	1
0 1 0	Not Master Mode	2
0 1 1	Parity Error	3
0 0 0	No Illegal Action	4

6) Execute Interrupt Line -- An unmasked Interrupt Cell in the Memory Controller will place a "1" on this line. It will set the corresponding flip-flop of the IPR-Register in the Control Frame.

10.1.2.2 Internal

The internal interface will be considered one frame at a time and are listed in Tables 10.1.2.2 and 10.1.2.2a.

Table 10.1.2.2 Interface Frame Inputs

Mnemonic	Description
Register Frame	
DDOOD-DD35D (ZDORF0-35)	DATA OUTPUT BUS
QDPPD($DP)	STROBE DOUBLE PRECISION
Arithmetic Frame	
FI09C (PAR MASK)	INDICATES PAR. ERR FAULT TRAP DISABLED
FI07C (TALLY RUNOUT)	LAST TALLY WAS ZERO
Associative Memory	
DC30E-DC35E(ZADS30-35)	ASSOC. REG. DESCRIPTOR BUS
BA55E(ZAU55)	ASSOC.REG.ADJUST USAGE BIT BUS
BE54E(ZEF54)	ASSOC.REG.EMPTY/FULL BIT BUS
BU56E-BU59E(ZSUC56-59)	SELCTD.USAGE COUNTER BUS
DA00E-DA17E(ZASM0-17)	ADDRESS MOD.64 BUS
DB18E(ZBB18)	BASE BIT BUS
DB19E-DB26E(ZABD19-26)	BOUNDARY BUS
DC27E-DC29E(ZADS27-29)	ASSOC.REG. DESCRIPTOR BUS
DT36E-DT53E(ZPTR50-53)	ASSOC.POINTER BUS
QNIAE($NIA)	NO INFORMATION AVAILABLE PULSE
QSSFE($SF)	START FETCH PULSE
QSULE($SET FA PMLP)	SET UP FAPM (LP) FETCH PULSE
QSUPE($SET PTW)	SET UP PTW FETCH PULSE
QSUSE($SET FA PMSP)	SET UP FAPM (SP) FETCH PULSE
QWBCE($START WBC)	START WRITTEN BIT CYCLE

Table 10.1.2.2 Interface Frame Inputs - contd

Mnemonic	Description
Control Frame	
BEADB(EA)	PREPARED ADDRESS IS FOR OPERAND
BFLSB(RPTS)	REPEAT FLAG DECODE FD OR FT
BIPCB(PC)	PREPARE INTERRUPT
BISIB	INHIBIT STROBE INTERRUPT
BMPAB(FPA)	FAULT OR EXECUTE INT. IN PA CYCLE
BSFTB	INHIBIT STROBE IF
BSPAB(CSET PA)	LEVEL TO SET PA
BSPIB(CSET PI)	LEVEL TO SET PI
BSPNB(CSET PN)	LEVEL TO SET PN
BSPRB(CSET PR)	LEVEL TO SET PR
BSPTB(CSET PT)	LEVEL TO SET PT
BSWNB(CSET WN)	LEVEL TO SET WN
BTBCB	WF AND STOP
BTRGB(TRG01)	TRANSFER GROUP SUB 1 DECODE
BYCFB	DIRECT TYPE OPERATION
DD00B-DD35B(ZDOCF0-35)	CONTROL FRAME DATA OUTPUT BUS
DW00B-DW17B(ZI0-17)	INST.REG.DECODE BUS
ECICB	CONNECT
ECT0B-ECT5B(CT0-5)	TAG BIT STORAGE REGISTER
EMLSB	MULTIPLE LOAD STORE
EMSYB(MSY)	MASTER/SLAVE FLAG
EPR0B-EPR7B(IPR0-7)	INTERRUPT PRIORITY REGISTER
EXSFB(XSF)	EXECUTE SWITCHES FAULT
MB00B-MB17B(BS0-17)	ADDRESS BASE ADDER
QAAMB($AAM)	START ASSOCIATIVE APPENDING STROBE
QCFFB($CFF)	COMPARE STROBE
QESRB	EA AND STROBE R
QPRCB($PRC)	PRE STROBE C
QSABB($SAB)	START ABSOLUTE STROBE
QSAPB($SAP)	START APPENDING STROBE
QSCAB($SCA)	STROBE TO SCANNER LOGIC
QZCCB($C)	GENERAL STROBE
QZDPB($DP)	DOUBLE PRECISION STROBE

Table 10.1.2.2 Interface Frame Inputs - contd

Mnemonic	Description
Base Frame	
BAB4F	PH.4 SIG. TO PORTS A&B INTERLACE
BABIF	PH.2 OR 4 SIG. TO PORTS A&B INTERLAC
BCD4F	PH.4 SIG. TO PORTS C&D INTERLACE
BCDIF	PH.2 OR 4 SIG TO PORTS C&D INTERLACE
BEF4F	PH.4 SIG. TO PORTS E&F INTERLACE
BEFIF	PH.2 OR 4 SIG. TO PORTS E&F INTERLAC
BGH4F	PH.4 SIG. TO PORTS G&H INTERLACE
BGHIF	PH.2 OR 4 SIG. TO PORTS G&H INTERLAC
BNPFF	PARITY ERROR
BP04F($\overline{\text{GP4}}$)	RECOGNIZE BUFFR.FILLING PI PARITY FAULT
BPETF(PRB=TBR)	CONTENTS OF PBR EQUALS CONTENTS OF TBR
BPIEF	PARITY (SPECIAL)
BPSIF	INHIBIT STROBE PARITY
BRPAF-BRPHF($\overline{\text{PORT ON}}$)	A-H PORT ON
BSFFF($\overline{\text{SFF}}$)	STOP FOR FAULTS
DD00F-DD35F(ZDOBF0-35)	36-BIT BASE FRAME DATA OUT BUS
DI00F-DI35F(ZDI0-35)	DATA INPUT BUS
DX00F-DX17F($\overline{\text{ZBRX0-17}}$)	BASE REGISTER EXTERNAL BUS
EGB0F(GB0)	GENERAL BASE INSTRUCTION F/F
EN03F-EN05F(INC3-5)	INCREMENTOR REGISTER
ERSTF	RESET
ESN0F(SN0)	SNAPSHOT ZERO F/F
ESN1F(SN1)	SNAPSHOT ONE F/F
EWF0F(WF)	WAIT FOR FAULT
FS00F-FS28F($\overline{\text{DSBR0-28}}$)	29-BIT DESCRIPTOR SEG.BASE REGISTER
QBRYF($BRY)	BASE READY PULSE
QDPBF($DP)	DOUBLE PRECISION STROBE
ZSN0F($SN0)	SNAPSHOT ZERO PULSE
QSNZF($SN1)	SNAPSHOT ONE PULSE

Table 10.1.2.2 Interface Frame Inputs - contd

Mnemonic	Description
Maintenance Panel	
SA00T-SA05T	ABSOLUTE ADDRESS SWITCH
SAMST	ASSOCIATE MEMORY OFF
SIHFT	INHIBIT FAULT SWITCH
SSCUT	STEP CU SWITCH
SSMMT	STEP MEMORY SWITCH
SSPGT	STEP PROGRAM SWITCH
SSTPT	STEP SWITCH
STSTT	TEST SWITCH

Table 10.1.2a Interface Frame Outputs

Arithmetic Frame

Mnemonic	Description
EHTAM(HTAL)	CHANGE TALLY RUNOUT IND
QDSTM($DS)	DATA STORED PULSE
QZPRM($PARITY)	PARITY PULSE

Associative Memory Frame

Mnemonic	Description
BMSTM	DECODE FOR STORE OR RAR CYCLES TO ASSOC. MEM.
BPTLM(WPTWL)	WRITE PTW LARGE
BPTSM(WPTWS)	WRITE PTW SMALL
BSDWM(WSDW)	WRITE SDW
EWBSM	SET WRITTEN AND USED BIT
FD00M-FD35M(DES0-35)	36-BIT DESCRIPTOR REGISTER
QDAMM($DAAM)	DATA AVAILABLE PULSE
QRELM($RELEASE)	RELEASE PULSE

Base Frame

BFAAM	$\overline{FA}$	FINAL ADDRESS FETCH(FAPM+FANP+ABS)
EIFAM-EIFCM	IARA-C	3-BIT ILLEGAL ACTION REG.
EP20M-EP27M	DIR20-7	PORT SELECTION F/F
EPINM		PREPARE INTERRUPT
ESPFM		STEP
ESSIM		STOP STROBE INTERRUPT
FD30M-FD32M		DES REGISTER
QAVFM	$AVF	ACCESS VIOLATION FAULT PULSE
QBORM	$BORY	START BASE OPERATION PULSE
QDFCM	$DF	DIRECTED FAULT PULSE
QDSTM	$DS	DATA STORED PULSE
QIASM	$SIAL	SPECIAL ILLEGAL ACTION PULSE
QICFM	$ICF	ILLEGAL CLASS FAULT PULSE
QINTM	$INT	STROBE INTERRUPT
QOBFM	$OBF	OUT OF BOUNDS FAULT PULSE
QOCNM	$CON	CONNECT PULSE FROM ALL PORTS
QOIAM	$IAL	ILLEGAL ACTION PULSE FROM ALL PORTS
QPINM	$PIN	STROBE PIN
QRSFM	$RSF	IF READY TO FAULT PULSE
QSDAM	$DA	DATA AVAILABLE PULSE

Table 10.1.2.2a Interface Frame Outputs - contd

Mnemonic	Description
Control Frame	
BDPCM	DOUBLE PRECISION COMMAND
BFAAM($\overline{\text{FA}}$)	FINAL ADDRESS FETCH(FAPM+FANP+ABS)
BLSRM	END MULTIPLE INST.
BIA0M-BIA3M($\overline{\text{IBA0-3}}$)	INTERRUPT BLOCK ADDRESS BITS 0-3
BRMLM(CRESET MLSF)	RESET MLSF F/F
DAIPM-DHIPM($\overline{\text{IP0-7}}$)	EXTERNAL INTERRUPT PRESENT-PORTS A-H
EMS2M(MSF2)	NEXT INST.PR.MASTER/SLAVE MODE BUFFE
ERDYM($\overline{\text{SRDY}}$)	PROC.READY FOR STEP FLAG
FN00M-FN17M(SCA0-17)	SCANNER BITS 0-17
MA15M-MA17M(RS15-17)	RS-ADDER BITS 15-17
QDSTM($DS)	DATA STORED PULSE
QIAZM($IAZ)	STROBE ILLEGAL ACTION
QIFFM($IFF)	INTERFACE FRAME FINISHED PULSE
QINTM ($INT)	STROBE INTERRUPT
QOPNM($PIN)	STROBE PIN
QPRIM($PRI)	STROBE C
QSDAM($DA)	STROBE DATA AVAILABLE
QSRFM($RSRF)	APPNDG.FETCH STARTED PULSE
QSSFM($SSRF)	FINAL FETCH STARTED PULSE
Maintenance Panel	
EABSM(ABS)	ABSOLUTE MEMORY FETCH CYCLE
EAMSM(AMS)	ASSOC.MEMORY SEQUENCE FLAG
EAPEM	APPEND PARITY ERROR
EBFRM(BFR)	BASE FRAME READY F/F
EBIFM(BIF)	BASE INSTRUCTION SPECIAL CONTROL F/F
ECFRM(CFR)	CONTROL FRAME READY F/F
EDAFM(DAF)	DATA AVAILABLE/STORED FLAG
EDDFM(DDAF)	DOUBLE PRECISION FLAG
EDINM(DPIN)	DOUBLE PRECISION PIN FLAG
EDPBM	DESCRIPTOR PAGE TABLE WORD
EDPTM(DSPTW)	DESC.SEG.PG.TBL.WD.MEMORY FETCH CYCL
EFANM(FANP)	FINAL ADDRESS(NON-PAGED MEMORY FETCH CYCLE
EFAPM(FAPM)	FINAL ADDRESS(PAGED)MEMORY FETCH CYCLE
EPTBM	PAGE TABLE WORD
ESDBM	SEGMENT DESCRIPTOR WORD

Table 10.2.2a Interface Frame Outputs - contd

Mnemonic	Description
Maintenance Panel	
EHTAM(HTAL)	TALLY RESULT HOLDING F/F (ZERO/ NON-ZERO)
EIFRM(IFR)	INTERFACE FRAME READY F/F
ELPFM(LPF)	PAGE SIZE FLAG
EMLSM(MLSF)	MULTIPLE LOAD/STORE INST.FLAG
EMS2M(MSF2)	NEXT INST.PR.MASTER/SLAVE MODE BUFFER
EMSFM(MSF1)	NEXT INST.PR.MASTER/SLAVE MODE F/F
EPINM(PIN)	MEMORY READY F/F
EPTWM(PTW)	PG.TBL.WD.MEMORY FETCH CYCLE
ESDNM(SDWN)	SEG.DESC.WD.FETCH-NONPAGED DESC.SEGMENT
ESDPM(SDWP)	SEG.DESC.WD.FETCH-PAGED DESC.SEGMENT
ESWUM(SWU)	SET WRITTEN/USED BIT CYCLE
EWAWM(WAW)	WAIT FOR APPENDING WORD FLAG
EWBFM(WBF)	SET WRITTEN/BIT FLAG(IF DETECTED)
EWBSM(WBS)	SET WRITTEN BIT SEQUENCE FLAG (IF DETECTED)
Relay Box	
EA00M-EA23M(ADR0-23)	24-BIT ADDRESS REGISTER
EIFAM-EIFCM(IAR0-2)	3-BIT ILLEGAL ACTION REGISTER
EP10M-EP17M(DIR1 0-7)	PORT SELECTION
EP20M-EP27M(DIR2 0-7)	PORT SELECTION
EZ00M(ZONE0)	ZONE REGISTER BIT 0
EZ02M(ZONE2	ZONE REGISTER BIT 2
EZ03M(ZONE3)	ZONE REGISTER BIT 3
EZ05M(ZONE5)	ZONE REGISTER BIT 5
EZL1M(ZONE L1)	ZONE REGISTER BIT L1
EZL4M(ZONE L4)	ZONE REGISTER BIT L4
EZU1M(ZONE U1)	ZONE REGISTER BIT U1
ECMAM-ECMDM	COMMAND REG.

Table 10.1.2.2a Interface Frame Outputs-contd

Mnemonic	Description
	Relay Box
EZU4M(ZONE U4)	ZONE REGISTER BIT U4
FD00M-FD35M(DES0-35)	36-BIT DESCRIPTOR REGISTER
F000M-F071M(DOR0-71)	72-BIT DATA OUTPUT REGISTER
MA00M-MA17M(RS0-17)	SUM OUTPUTS OF RS ADDER

10.1.3 ADDRESS PREPARATION

The Interface Frame's function in address preparation is to take the address formed by the Control Frame and convert it to the address required by the Memory Controller. This can be done in several ways depending upon the mode of operation. In general, the Interface Frame takes the address from the Control Frame and using information obtained from the Base Frame, Associative Memory, and Core Memory as required, forms the final address that is presented to the Memory Controller. This discussion will deal strictly with the portion of the Interface Frame operation from the initiation of an address preparation cycle to the issuance of a $INT for an operand or instruction pair. This does not include any address modification that takes place in the Control Frame, or any of the other functions of the Interface Frame. The three modes of operation, each started by a pulse from the Control Frame, are:

1) Absolute - started by $SAB.

2) Append Without Associative Memory - started by $SAP.

3) Append With Associative Memory - started by $AAM.

See charts and timing diagrams at end of this section.

10.1.3.1 Absolute Mode

The absolute mode of operation is characterized by the Interface Frame accepting the address presented by the Control Frame as the final address. If it is for an operand or instruction it is also considered a final effective address.

10.1.3.1 - contd

After the Control Frame has embarked on a prepare address cycle, it will send a $SAB to the Interface Frame. This pulse will set the ABS flip-flop and enter a delay line to emerge as $IFR.. During the time interval between ABS set and $IFR the following events occur:

1) FA becomes true.

2) (EA) (FA) becomes true if this is a final effective address for an operand.

3) ¢BS_{0-11}/RS_{6-17}, ¢BS_{12-17}/ADR_{18-23}, ¢RS_{0-17}/ADR_{0-17} all come true switching the output of the BS-adder with six leading zeros to the input to the Address Register (ADR_{0-23}).

4) $IFF will be sent to the Control Frame to set its IFR flip-flop.

$IFR will set the Interface Frame Ready flip-flop in the Interface Frame. When the Base Frame and the Control Frame become ready, (in this case it will happen at the same time), the BFR and CFR flip-flops will set. At this time a last pulse detector will generate a $A, which will:

1) Be delayed and then generate a $ALS.

2) Generate a $INT.

3) Reset BFR and CFR.

4) Generate $ADR.

$ALS will reset IFR, and ABS.

$ADR will strobe the address into the Address Register.

10.1.3.1 - contd

$INT is sent to the selected Memory Controller and it will:

1) Generate $DOR if required.

2) Reset PIN.

This completes the absolute mode.

10.1.3.2 Append Without Associative Memory

This mode of operation is characterized by the Interface Frame taking the address presented by the Control Frame and appending it to a nonpaged SDW or to a PTW to form the final address. All intermediate steps deal with information from core memory and registers in the Base Frame. This mode is only used if a store Associative Memory operation is in progress or if the TEST ENABLE and ASSOC MEM OFF switches on the Maintenance Panel are enabled. There are many possible combinations of ways to arrive at a final address. The shortest and longest will be discussed.

Assume that the Segment Descriptor Table is not paged and that an instruction pair is to be fetched from a nonpaged segment. After the Control Frame has embarked on a prepare address cycle it will send a $SAP to the Interface Frame. ($DSBR_{27}$ and $DSBR_{28}$ both = 1, SDW_{27} and SDW_{28} both = 1).

10.1.3.2 - contd

$SAP will generate a $ Begin, which will set SDWN, send a $RSRF to the Control Frame, and enter a delay line to emerge as $IFR. During the time interval between SDWN set and $IFR the following events occur:

1) SDC comes true.

2) ¢$DSBR_{19-26}$/BC_{4-11} and ¢$ZBRX_{0-11}$/BC_{0-11} come true allowing the boundary check to be made.

3) ¢$DSBR_{0-17}$/RS_{0-17}, ¢$ZBRX_{0-11}$/RS_{6-17}, ¢RS_{0-17}/ADR_{0-17}, and ¢$ZBRX_{12-17}$/ADR_{18-23} all come true causing the page and word number in the procedure Base Register to be added to the DSBR producing the address, in core memory, of the SDW.

$IFR will set the IFR flip-flop in the Interface Frame. At this time SDC, PIN, and IFR will cause a last pulse detector to generate a $A, which will:

1) Generate a $ALS, which will reset IFR in the Interface Frame.

2) Generate a $INT if there isn't an Out of Bounds Fault, or an Access Violation.

3) Generate a $ADR, which will strobe the address into ADR_{0-23}

$INT is sent to the selected Memory Controller and it will:

1) Reset PIN.

2) Set WAW.

10.1.3.2 - contd

The Interface Frame will now wait until the Memory Controller acknowledges the request by sending a \$PIN. This pulse will be sent on to the Control Frame and delayed before it sets PIN, and is sent on to the Base Frame. When the Memory Controller has placed the SDW on the ZDI lines it will send a \$DA. This pulse will be forwarded on to the Base and Control Frames and it will enter a delay line to emerge and generate \$I, \$IS, and \$DES.

\$I samples the reset side of all the appending control flip-flops. In this case it will reset WAW, SDWN, and LPF.

\$IS samples the set side of all the appending control flip-flops. In this case it will set FANP, and it will issue a \$FAN, which is sent to the Control Frame as \$SSRF. It will also enter a delay line to emerge as \$IFR, which will set IFR.

\$DES will strobe the SDW on the ZDI lines into the Descriptor Register.

The ZDI_{29-35} lines are routed through the Descriptor Generator. However, in this case no changes will be made to the descriptor field of the SDW before it is strobed into DES_{29-35}. The descriptor field on the ZDI lines will be checked for a directed fault, and as soon as FANP is set, the descriptor field in the Descriptor Register will be checked for class faults.

10.1.3.2 - contd

During the time interval between FANP set and $IFR the following events occur:

1) $IFF will be sent to the Control Frame to set its IFR flip-flop.

2) SDC goes false.

3) FA will come true.

4) ¢BS_{0-11}/BC_{0-11} and DES_{19-26}/BC_{4-11} come true allowing the boundary check to be made.

5) ¢DES_{0-17}/RS_{0-17}, ¢BS_{0-11}/RS_{6-17}, ¢$RS_{0\div17}$/ADR_{0-17}, and ¢BS_{12-17}/ADR_{18-23} all come true causing the word number of the effective address to be added to the SDW producing the address, in core memory, of the instruction pair.

6) $IFR will set the IFR flip-flop in the Interface Frame.

When the three related frames are ready, as signified by BFR, CFR, and IFR being set, FA is true, and PIN is set, a last pulse detector will generate a $A, which will:

1) Generate a $ALS, which will reset IFR and FANP.

2) Generate a $INT if there isn't an Out of Bounds fault or AVF.

3) Reset BFR and CFR.

4) Generate a $ADR which will strobe the address into ADR_{0-23}.

10.1.3.2 - contd

$INT is sent to the selected Memory Controller and it will:

1) Reset PIN.

The Interface Frame will now wait until the Memory Controller acknowledges the request. Since this is a request for an instruction pair the Memory Controller will send two $PINs and two $DAs. The instruction pair will be placed on the ZDI lines. The $PINs and the $DAs will be forwarded on to the Base and Control Frames. This completes this particular operation.

Assume that it is not a Base instruction, the Segment Descriptor Table is paged and that an operand is to be stored in a large page. ($DSBR_{27}$="1", $DSBR_{28}$="0", SDW_{27}="0" and SDW_{28}="0" the base bit is on). After the Control Frame has embarked on a prepare address cycle, it will send a $SAP to the Interface Frame.

$SAP will generate a $ Begin, which will set DSPTW, send a $RSRF to the CF, and enter a delay line to emerge as $IFR. During the time interval between DSPTW set and $IFR the following events occur:

1) SDC comes true.

2) ¢$ZBRX_{0-11}/BC_{0-11}$ and ¢$DSBR_{19-26}/BC_{4-11}$ come true allowing the boundary check to be made.

3) ¢$DSBR_{0-17}/RS_{0-17}$, ¢$ZBRX_{4-11}/RS_{16-17}$, ADR_{18-23}, and ¢RS_{0-17}/ADR_{0-17} come true causing the page number in the Procedure Base Register to be added to the DSBR producing the address, in core memory, of the DSPTW.

10.1.3.2 - contd

$IFR will set the IFR flip-flop in the Interface Frame. At this time SDC, PIN, and IFR will cause a last pulse detector to generate a $A, which will:

1) Generate a $ALS, which will reset IFR in the Interface Frame.

2) Generate a $INT if there isn't an Out of Bounds Fault, or AVF.

3) Generate a $ADR that will strobe the address into ADR_{0-23}.

$INT is sent to the selected Memory Controller and it will:

1) Reset PIN.

2) Set WAW.

The Interface Frame will now wait until the Memory Controller acknowledges the request by sending a $PIN. This pulse will be sent on to the Control Frame and delayed before it sets PIN and is sent on to the Base Frame. When the Memory Controller has placed the DSPTW on the ZDI lines it will send a $DA. This pulse will be sent on to the Base and Control Frames and it will enter a delay line to emerge and generate $I, $IS, and $DES.

$I samples the reset side of all the appending control flip-flops. In this case it will reset DSPTW and WAW.

$IS samples the set side of the appending control flip-flops. In this case it will set SDWP and enter a delay line to emerge as $IFR, which will set IFR.

10.1.3.2 - contd

$DES will strobe the DSPTW on the ZDI lines into the Descriptor Register.

The ZDI_{29-35} lines are routed through the Descriptor Generator. However, in this case, no changes will be made to the descriptor field of the DSPTW before it is strobed into DES_{29-35}. The descriptor field on the ZDI lines will be checked for a directed fault.

During the time interval between SDWP set and $IFR the following event occurs:

1) ¢$ZBRX_{12-17}$/ADR_{18-23}, ¢DES_{0-17}/RS_{0-17}, and ¢RS_{0-17}/ADR_{0-17} will come true concatenating the word number in the Procedure Base Register to the DSPTW in the Descriptor Register producing the address, in core memory, of the SDW.

$IFR will set IFR in the Interface Frame. At this time SDC, PIN, and IFR will cause a last pulse detector to generate a $A, which will:

1) Generate a $ALS, which will reset IFR.

2) Generate a $INT.

3) Generate a $ADR that will strobe the address into ADR_{0-23}.

The Interface Frame will now wait until the Memory Controller acknowledges the request by sending a $PIN. This pulse will be sent on to the Control Frame and delayed before it sets PIN and is sent on to the Base Frame. When the Memory Controller has placed the SDW on the ZDI lines it will send a $DA. This pulse will be sent on to the Base and Control Frames and it will enter a delay line to emerge and generate $I, $IS, and $DES.

10.1.3.2 - contd

$I samples the reset side of the appending control flip-flops. In this case, it will reset SDWP and WAW.

$IS samples the set side of the appending control flip-flops. In this case it will set PTW and LPF and enter a delay line to emerge as $IFR.

$DES will strobe the SDW on the ZDI lines into the Descriptor Register. The descriptor field will not be altered, but it will be checked for a directed fault or an illegal class fault.

During the time interval between PTW set and $IFR the following events occur:

1) ¢BS_{0-7}/BC_{4-11} and ¢DES_{19-26}/BC_{4-11} will come true allowing the boundary check to be made.

2) ¢DES_{0-17}/RS_{0-17}, ¢RS_{0-17}/ADR_{0-17}, and ¢BS_{0-7}/RS_{16-17}, ADR_{18-23} will come true concatenating the page number from the effective address to the SDW producing the address, in core memory of the PTW.

$IFR will set IFR. At this time SDC, PIN and IFR will cause a last pulse detector to generate a $A, which will:

1) Generate a $ALS which will reset IFR.

2) Generate a $INT if there isn't an Out of Bounds Fault, or AVF.

3) Generate a $ADR, which will strobe the address into ADR_{0-23}.

10.1.3.2 - contd

$INT is sent to the selected Memory Controller and it will:

1) Reset PIN.

2) Set WAW.

The Interface Frame will now wait until the Memory Controller acknowledges the request by sending a $PIN. This pulse will be sent on to the Control Frame and delayed before it sets PIN and is sent on to the Base Frame. When the Memory Controller has placed the PTW on the ZDI lines, it will send a $DA. This pulse will be sent on to the Base and Control Frames, and it will enter a delay line to emerge and generate $I, $IS, and $DES.

$I samples the reset side of the appending control flip-flops. In this case it will reset PTW and WAW.

$IS samples the set side of the appending control flip-flops. In this case it will set FAPM and enter a delay line to emerge as $IFR.

$DES will strobe the PTW on the ZDI lines into the Descriptor Register, In this case a descriptor field will be generated for the PTW by the Descriptor Generator before it is strobed into DES_{27-35}. (See paragraph 10.1.4.2 on the Descriptor Generator for a description of this operation.) The descriptor field will be checked for directed faults, illegal class faults, or access violations.

10.1.3.2 - contd

During the time interval between FAPM set and $IFR the following events occur:

1) $IFF will be sent to the Control Frame to set its IFR flip-flop.

2) SDC goes false.

3) FA and EA will come true.

4) ¢$\mathrm{DES}_{0-17}/\mathrm{RS}_{0-17}$, ¢$\mathrm{PS}_{0-13}/\mathrm{ADR}_{0-13}$ and $\mathrm{BS}_{8-11}/\mathrm{ADR}_{14-17}$, and ¢$\mathrm{BS}_{12-17}/\mathrm{ADR}_{18-23}$ will come true concatentating the word number of the effective address to the PTW producing the address, in core memory, of the location in which the operand is to be stored.

5) $IFR will set IFR.

When the three related frames are ready, as signified by BFR, CFR, and IFR set, FA true, and PIN set a last pulse detector will generate a $A, which will:

1) Generate $ALS, which will reset FAPM and IFR.

2) Generate a $INT.

3) Reset BFR and CFR.

4) Generate a $ADR, which will strobe the address into ADR_{0-23}.

10.1.3.2 - contd

$INT is sent to the selected Memory Controller and it will:

1) Reset PIN.

2) Generate a $DS, which is sent to the AF, CF, and BF.

The operand will be stored in the address specified by the Address Register, thus completing this operation.

10.1.3.3 Append with Associative Memory

This mode of operation is characterized by the Interface Frame being controlled by the Associative Memory. It will set up for a particular type of operation, and then it will proceed with it or change to a new type upon command of the Associative Memory. This will be the mode of operation normally used.

The Associative Memory can store up to 16 SDWs or PTWs. At the start of an appending cycle it is searched for the required SDW or PTW. If either one is found, it will be used eliminating the need for a core memory access. If the required SDW or PTW is not found it will be obtained from core memory and stored in the Associative Memory, where it can be accessed the next time it is needed. There are five possible outcomes of an Associative Memory search, plus an additional special case involving the written bit. They are:

1) No information available.

2) No information available after a pointer match.

3) Nonpaged SDW found.

4) Paged SDW found.

5) PTW found.

See section 11.0 on the Associative Memory for the details of the search operation and the development of the various pulses. In the following discussions it will be assumed that all appending control flip-flops are reset at the start of the cycle, that a $AAM has been received from the Control Frame setting AMS and FANP, and that the Associative Memory has started its search.

10.1.3.3 - contd

1) If the search is unsuccessful the Associative Memory will send a $NIA which will:

   a) Reset AMS.
   b) Generate a $ RESTART.
   c) Generate a $ BEGIN.

   $ RESTART will reset FANP.

   $ BEGIN will start an appending cycle that is the same as started by a $SAP from the Control Frame, except that SDWs or PTWs retrieved from core memory will be stored in the Associative Memory.

2) If the search produces a pointer match (paged and no SDW present) the Associative Memory will send either a $PFLP (QSUL1) or a $PFSP (QSUS1). Either pulse will reset FANP and set FAPM. $PFLP will set LPF. $PFSP will reset LPF. If the search produces no further match the Associative Memory will send a $NIA, which will:

   a) Reset AMS.

   b) Generate a $ RESTART, which will reset FAPM.

   c) Generate a $ BEGIN.

   $ BEGIN will start an appending cycle that is the same as started by a $SAP from the Control Frame.

3) If the search produces a match with a nonpaged SDW (bit 28 = 1) (64 words) the Associative Memory will send a $GO (QSSF1). The Interface Frame is set up in the Final Address Nonpaged mode. $GO will reset LPF and enter a delay line to emerge and generate $IFR.

10.1.3.3 - contd

During the time interval between \$GO and \$IFR the following events occur:

a) FA, EA, and FAN will come true.

b) \$IFF is sent to the Control Frame to set its IFR flip-flop.

c) ¢BS_{0-11}/BC_{0-11} and ¢$ZABD_{19-26}$ and BC_{4-11} will come true allowing the boundary check to be made.

d) ¢$ZASM_{12-17}/ADR_{18-23}$ will come true causing the combined page and word number of the effective address to be added to the SDW, from the Associative Memory, producing the address of the operand.

\$IFR will set IFR.

From this point on the Interface Frame will proceed the same as it did when it did not use the Associative Memory. The first \$A will reset AMS.

4) If the search produces a match with a paged SDW (1024 words) the sequence of events will be as follows.

At the end of the pointer search the Associative Memory will send a \$PFLP (QSUL1), which will:

a) Reset FANP.

b) Set FAPM and LPF.

10.1.3.3 - contd

At the end of the PTW search, where no match is found, the Associative Memory will send a $GO (QSSF1), which will enter a delay line to emerge and generate $IFR.

During the time interval between $GO and $IFR the following events occur:

a) ¢BS_{0-7}/BC_{4-11} and ¢$ZABD_{19-26}/BC_{4-11}$ will come true allowing the boundary check to be made.

b) ¢$ZASM_{0-17}/RS_{0-17}$, ¢BS_{0-17}/RS_{16-17}, ADR_{18-23}, and ¢RS_{0-17}/ADR_{0-17} will come true adding the page number from BS to the SDW producing the address of the PTW.

$IFR will set IFR.

From this point on the Interface Frame will proceed the same as it did when it did not use the Associative Memory. The first $I will reset AMS.

5) If the search produces a match with a small PTW (64 words) the sequence of events will be as follows.

At the end of the pointer search the Associative Memory will send a $PFSP (QSUS1), which will:

a) Reset FANP.

b) Set FAPM.

At the end of the PTW search the Associative Memory will send a $GO (QSSF1) which will enter a delay line to emerge and generate a $IFR.

10.1.3.3 - contd

During the time interval between $GO and $IFR the following events occur.

a) FA, EA, and FAN will come true.

b) $IFF will be sent to the Control Frame to set its IFR flip-flop.

c) ¢ZASM$_{0-13}$/ADR$_{0-13}$, ¢ZASM$_{14-17}$/ADR$_{11-17}$, and ¢BS$_{12-17}$/ADR$_{18-23}$ will come true concatenating the word number from the effective address to the PTW from the Associative Memory, producing the address of the operand.

$IFR will set IFR.

From this point on the Interface Frame will proceed the same as it did when it did not use the Associative Memory. The first $A will reset AMS.

This concludes the address preparation except for special conditions which will be considered next.

10.1.4 SPECIAL OPERATION

Items to be discussed under this general heading are Written and Use bits, Descriptor Generator, Base Instructions, and Multiple Load Store Operation.

10.1.4.1 Written and Use Bits

It is necessary to be able to tell if a page in core memory has been used or if it has been altered by looking at its PTW. Two bits (25 and 26) are reserved for this purpose. If bit 25 (Use bit) is a "1" the page has been used. If bit 26 (Written bit) is a "1" the page has been altered. When a page is initially stored in core memory and its PTW created both bits will be a "0." The first time the page is accessed or written into the corresponding bit of the PTW in both core memory and the Associative Memory will be made a "1."

Assume that the Interface Frame has just issued a $INT for a PTW, it is a store operation, and bits 25 and 26 are both "0." The Memory Controller will place the PTW on the ZDI lines and then send a $DA. This pulse will be sent to the Base and Control Frames and it will enter a delayline to emerge and generate $I, $IS, and $DES.

$I samples the reset side of the appending control flip-flops. In this case it will reset PTW and WAW.

$IS samples the set side of the appending control flip-flops. In this case it will set SWU, WBF, and enter a delayline to emerge as $IFR.

$DES will strobe the PTW on the ZDI lines into the Descriptor Register. In this case a descriptor field will be generated for the PTW and bit 29 will be forced to a "1" by the Descriptor Generator before it is strobed into DES_{27-35}. The Descriptor field will be checked for directed faults, illegal class faults, and access violations.

10.1.4.1 - contd

During the time interval between SWU and $IFR the following events occur:

1) SDC goes false.
2) A clear-write command to core memory is prepared.
3) The Zone Register is set to 00000100, which will cause only bits 24, 25, and 26 to be altered.
4) A "1" is placed on data out lines 25 and 26.

$IFR will set IFR. At this time SWU, PIN, and IFR will cause a last pulse detector to generate a $A, which will:

1) Generate a $ALS, which will reset IFR.
2) Generate a $INT.
3) Generate a $ADR, which is inhibited from strobing ADR_{0-23} by SWU.

$INT is sent to the selected Memory Controller and it will:

1) Cause a "1" to be written into bits 25 and 26 of the PTW in core memory.
2) Reset PIN.
3) Set WAW.

The Interface Frame will now wait until the Memory Controller acknowledges the request by sending a $PIN. This pulse will be sent on to the Control Frame and delayed before it sets PIN and is sent on the Base Frame.

10.1.4.1- contd

When the Memory Controller has completed the clear-write operation it will send a $DA. This pulse will be sent to the Base and Control Frames, and it will enter a delay line to emerge and generate $I and $IS.

$I samples the reset side of the appending control flip-flops. In this case it will reset SWU, WBF, and WAW.

$ITS samples the set side of the appending control flip-flops. In this case it will set FAPM and enter a delay line to emerge as $IFR. From this point on it will continue the same as for any FAPM cycle.

The same cycle will be gone through if only bit 25 or bit 26 need to be made a "1."

One other possibility exists, and that is that a PTW with the Written bit a "0" is found in the Associative Memory during a store or RAR operation.

When a store or RAR operation is decoded, the Interface Frame places a "1" on the BMST1 line to the Associative Memory. If a PTW with the Written bit a "0" is found the Associative Memory will send the Interface Frame a $SWBC (QWBC1) instead of the $GO (QSSF1). $SWBC will set WBS and generate a $ RESTART.

WBS set will place a "1" on the BWSB1 line to the Associative Memory, which will cause it to make its Written bit a "1."

$ RESTART will reset FAPM and generate a $ BEGIN.

10.1.4.1 - contd

$ BEGIN will strobe DWPTW and SDWN. If the Descriptor Table is paged DSPTW will set, otherwise SDWN will set. Assume that SDWN set. $ BEGIN will also send a $RSRF to the Control Frame and enter a delay line to emerge as $IFR. During the time interval between SDWN set and $IFR the following events occur:

1) SDC come true.

2) ¢$DSBR_{19-26}/BC_{4-11}$ and $ZBRX_{0-11}/BC_{0-11}$ come true allowing the boundary check to be made.

3) ¢$DSBR_{0-17}/RS_{0-17}$, ¢$ZBRX_{0-11}/RS_{6-17}$, ¢RS_{0-17}/ADR_{0-17}, and ¢$ZBRX_{12-17}/ADR_{18-23}$ all come true causing the page and word number in the Procedure Base Register to be added to the DSBR producing the address of the SDW.

$IFR will set IFR. At this time SDC, PIN, and IFR will cause a last pulse detector to generate a $A, which will:

1) Generate a $ALS, which will reset IFR.

2) Generate a $INT if there isn't an Out of Bounds fault.

3) Generate a $ADR that will strobe the address into ADR_{0-23}.

$INT is sent to the selected Memory Controller and it will:

1) Reset PIN.

2) Set WAW.

10.1.4.1 - contd

The Interface Frame will now wait until the Memory Controller acknowledges the request by sending a $PIN. This pulse will be sent on to the Control Frame and delayed before it sets PIN and is sent on to the Base Frame. When the Memory Controller has placed the SDW on the ZDI lines it will send a $DA. This pulse will be sent on to the Base and Control Frames and it will enter a delay line to emerge and generate $I, $IS, and $DES.

$I samples the reset side of the appending control flip-flops. In this case it will reset SDWN and WAW.

$IS samples the set side of the appending control flip-flops. In this case it will set SWU and enter a delay line to emerge as $IFR.

$DES will strobe the SDW on the ZDI lines into the Descriptor Register. The descriptor field will not be altered but it will be checked for a Directed Fault or an Illegal Class Fault.

During the time interval between SWU and $IFR the following events occur:

1) SDC goes false.

2) A clear-write command to core memory is issued.

3) The Zone Register is set to 00000100, which will cause only bits 24, 25, and 26 to be altered.

4) A "1" is placed on data out lines 25 and 26.

10.1.4.1 - contd

$IFR will set IFR. At this time SWU, PIN, and IFR will cause a last pulse detector to generate a $A, which will:

1) Generate a $ALS, which will reset IFR.

2) Generate a $INT.

3) Generate a $ADR, which will strobe the address into the Address Register.

$INT is sent to the selected Memory Controller and it will:

1) Cause a "1" to be written into bits 25 and 26 of the PTW in core memory.

2) Reset PIN.

3) Set WAW.

The Interface Frame will now wait until the Memory Controller acknowledges the request by sending a $PIN. This pulse will be sent on to the Control Frame and delayed before it sets PIN and is sent on to the Base Frame. When the Memory Controller has completed the clear-write operation it will send a $DA. This pulse will be sent to the Base and Control Frames, and it will enter a delay line to emerge and generate $I and $IS.

$I samples the reset side of the appending control flip-flops. In this case it will reset SWU, WBS, and WAW.

$IS samples the set side of the appending control flip-flops. In this case it will set FAPM and enter a delay line to emerge as $IFR.

10.1.4.1 - contd

During the time interval between FAPM set and $IFR the following events occur:

1) $IFF will be sent to the Control Frame to set its IFR flip-flop.

2) FA and possibly EA will come true.

3) ¢ZASM$_{0-13}$/ADR$_{0-13}$, ¢ZASM$_{14-17}$/ADR$_{14-17}$, and BS$_{12-17}$/ADR$_{18-23}$ will all come true concatenating the word number of the effective address to the PTW from the Associative Memory to form the operand address.

$IFR will set IFR.

From this point on it will continue as any final address cycle.

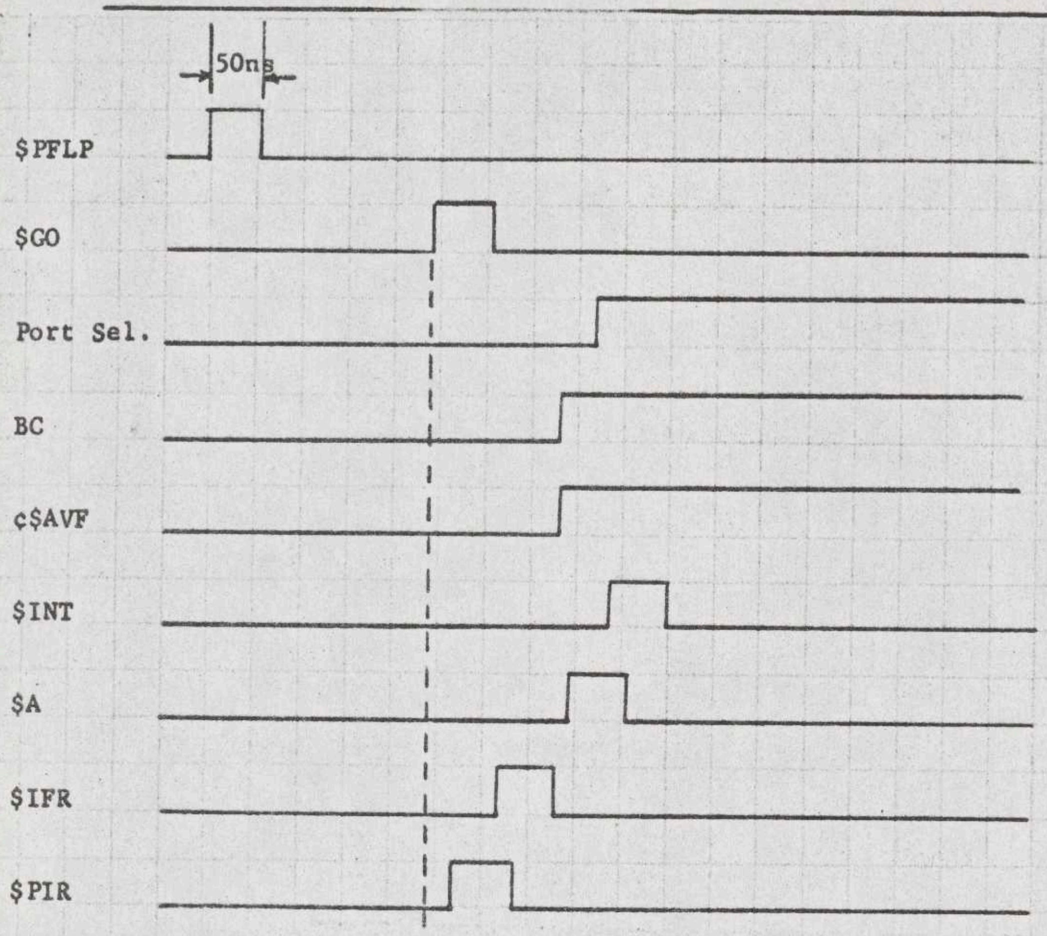

50ns
$PFLP
$GO
Port Sel.
BC
¢$AVF
$INT
$A
$IFR
$PIR

Fig. 10.1.4 FAPM (AMS) Mode, Timing Diagram

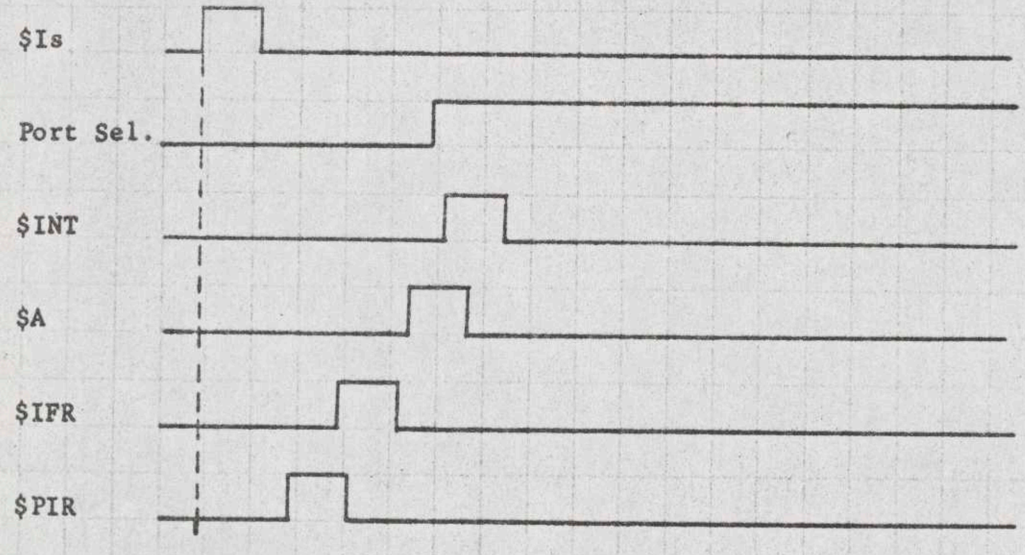

$Is
Port Sel.
$INT
$A
$IFR
$PIR

Fig. 10.1.4a SWU ($\overline{\text{WBS}}$) Mode, Timing Diagram

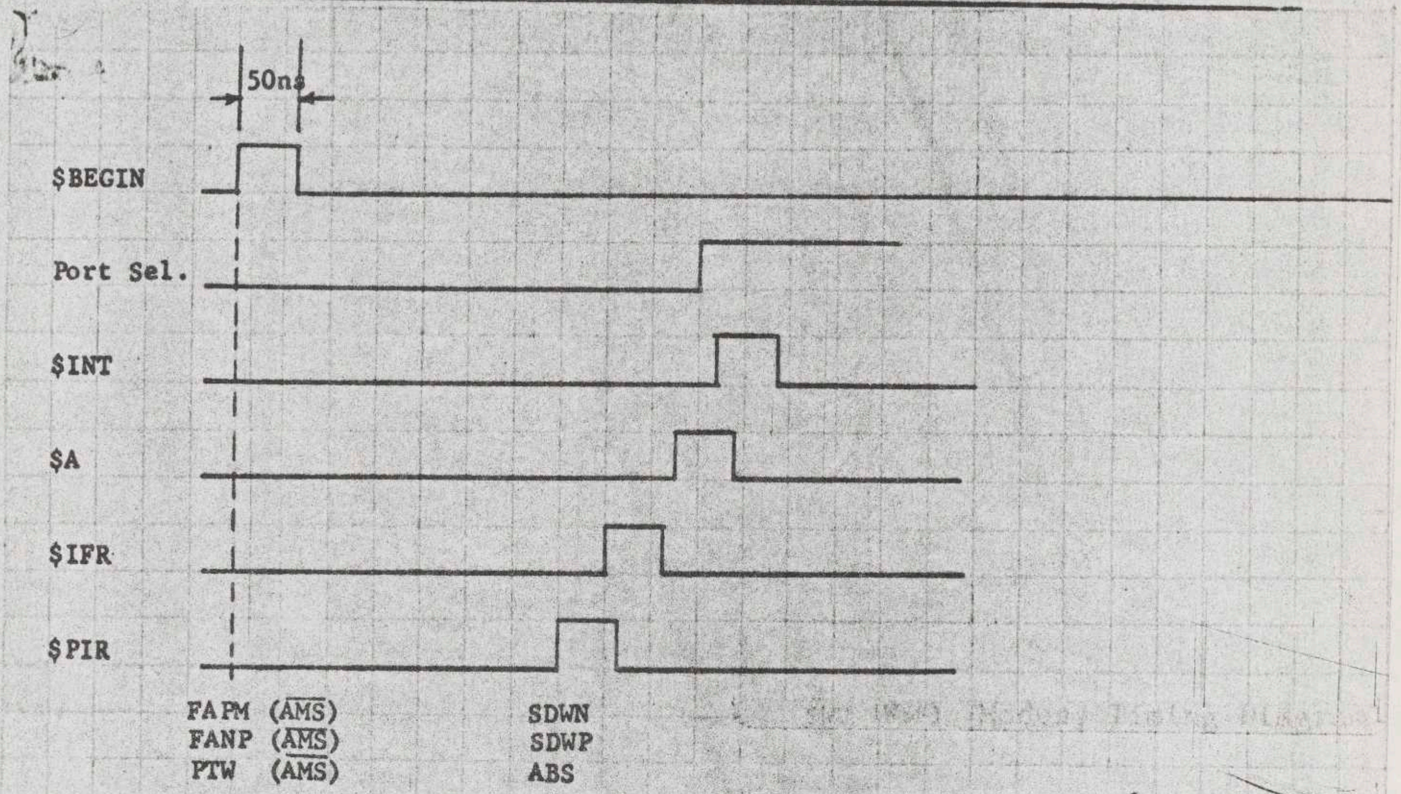

50ns
$BEGIN
Port Sel.
$INT
$A
$IFR
$PIR
FAPM ($\overline{\text{AMS}}$)
FANP ($\overline{\text{AMS}}$)
PTW ($\overline{\text{AMS}}$)
SDWN
SDWP
ABS

Fig. 10.1.4b SWU (WBS) Modes, Timing Diagram

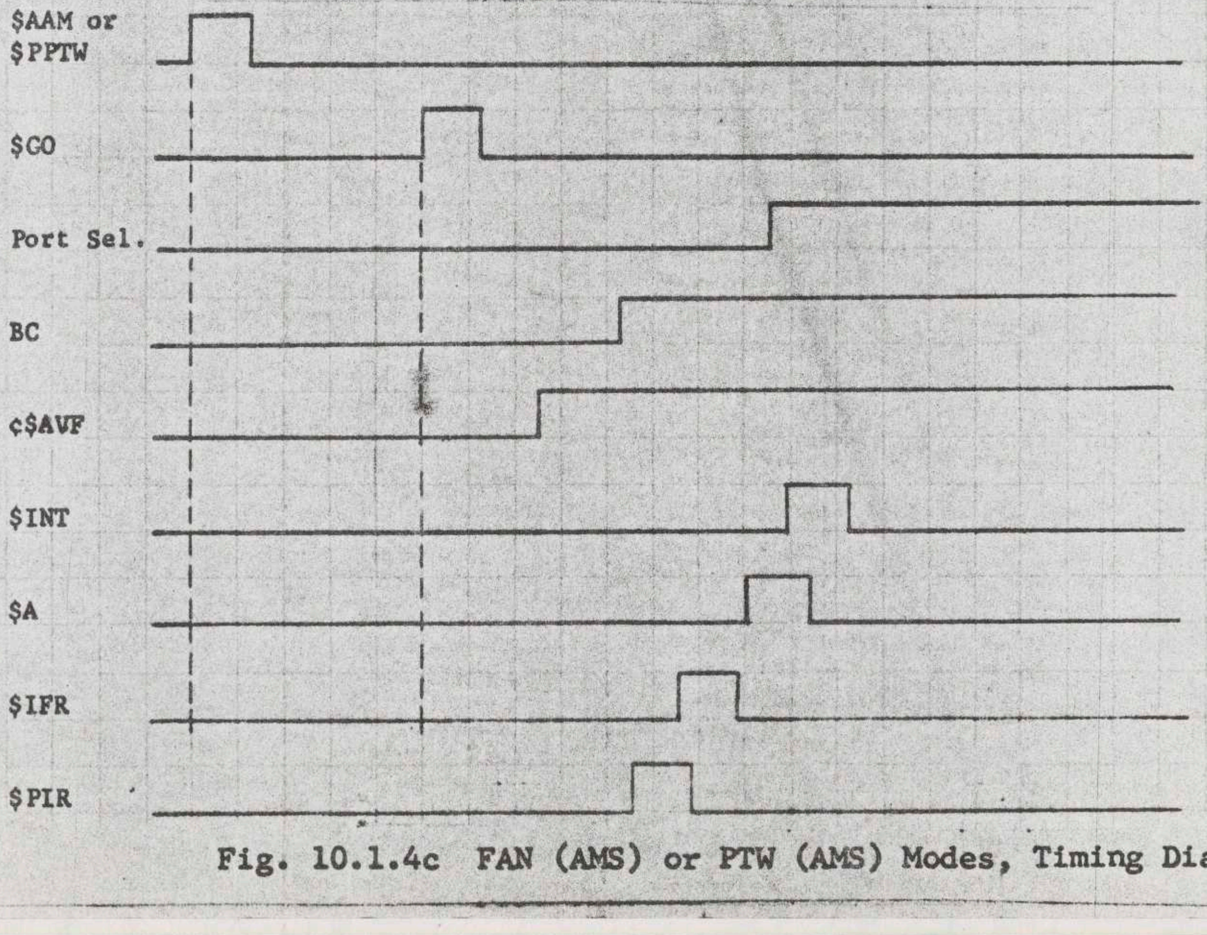
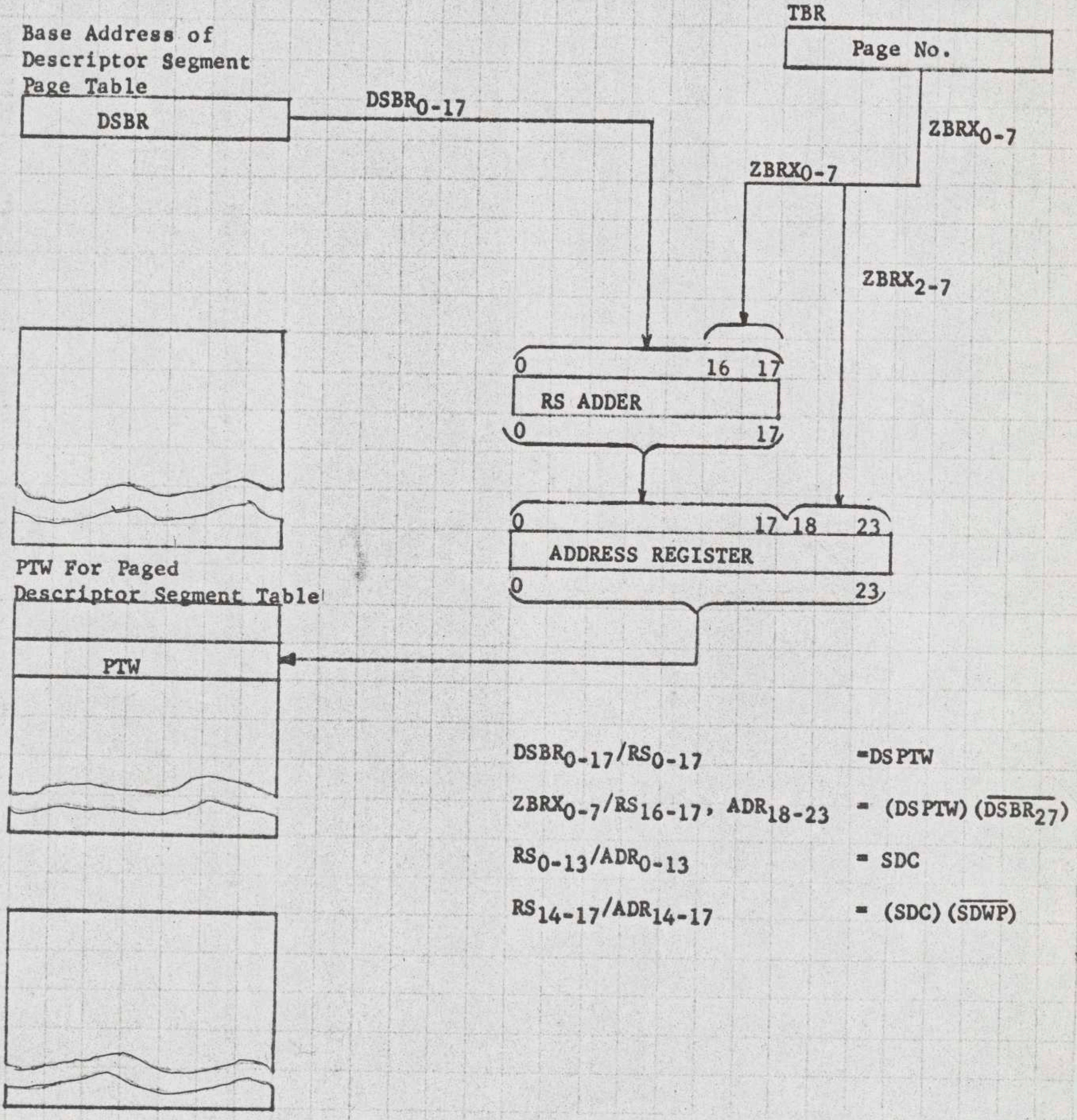

$AAM or
$PPTW
$GO
Port Sel.
BC
¢$AVF
$INT
$A
$IFR
$PIR

Fig. 10.1.4c FAN (AMS) or PTW (AMS) Modes, Timing Diagram


TBR
Page No.
Base Address of Descriptor Segment Page Table
DSBR
DSBR0-17
ZBRX0-7
ZBRX0-7
ZBRX2-7
0
16
17
RS ADDER
0
17
0
17
18
23
ADDRESS REGISTER
0
23
PTW For Paged Descriptor Segment Table
PTW

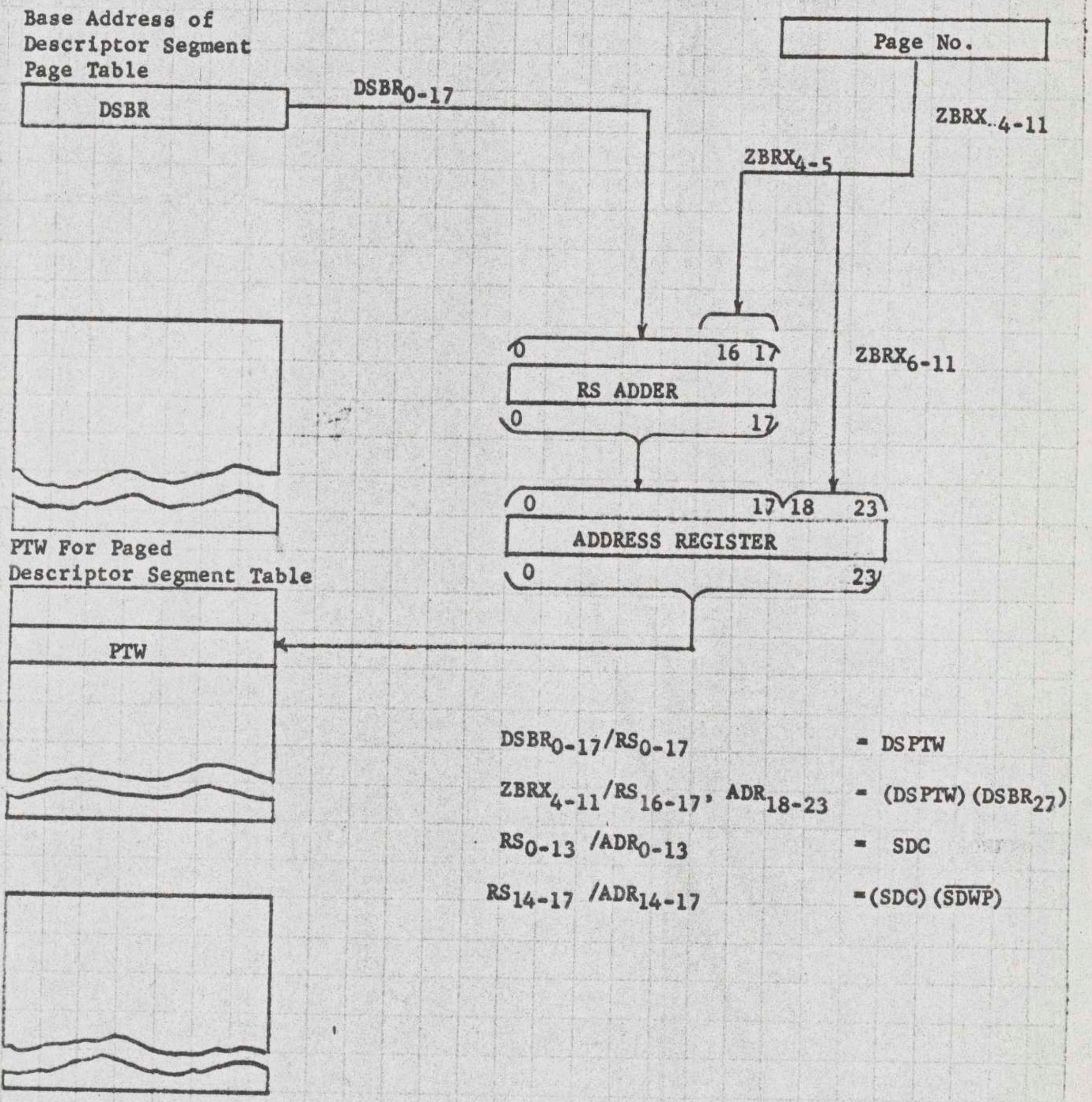
$DSBR_{0-17}/RS_{0-17}$ = DSPTW

$ZBRX_{0-7}/RS_{16-17}$, ADR_{18-23} = $(DSPTW)(\overline{DSBR_{27}})$

RS_{0-13}/ADR_{0-13} = SDC

RS_{14-17}/ADR_{14-17} = $(SDC)(\overline{SDWP})$

Fig. 10.1.4d Descriptor Segment Page Table Word Fetch - Large Page, Flow Chart


Base Address of Descriptor Segment Page Table
DSBR
$DSBR_{0-17}$
Page No.
$ZBRX_{4-11}$
$ZBRX_{4-5}$
$ZBRX_{6-11}$
0
16 17
RS ADDER
0
17
0
17 18
23
ADDRESS REGISTER
0
23
PTW For Paged Descriptor Segment Table
PTW

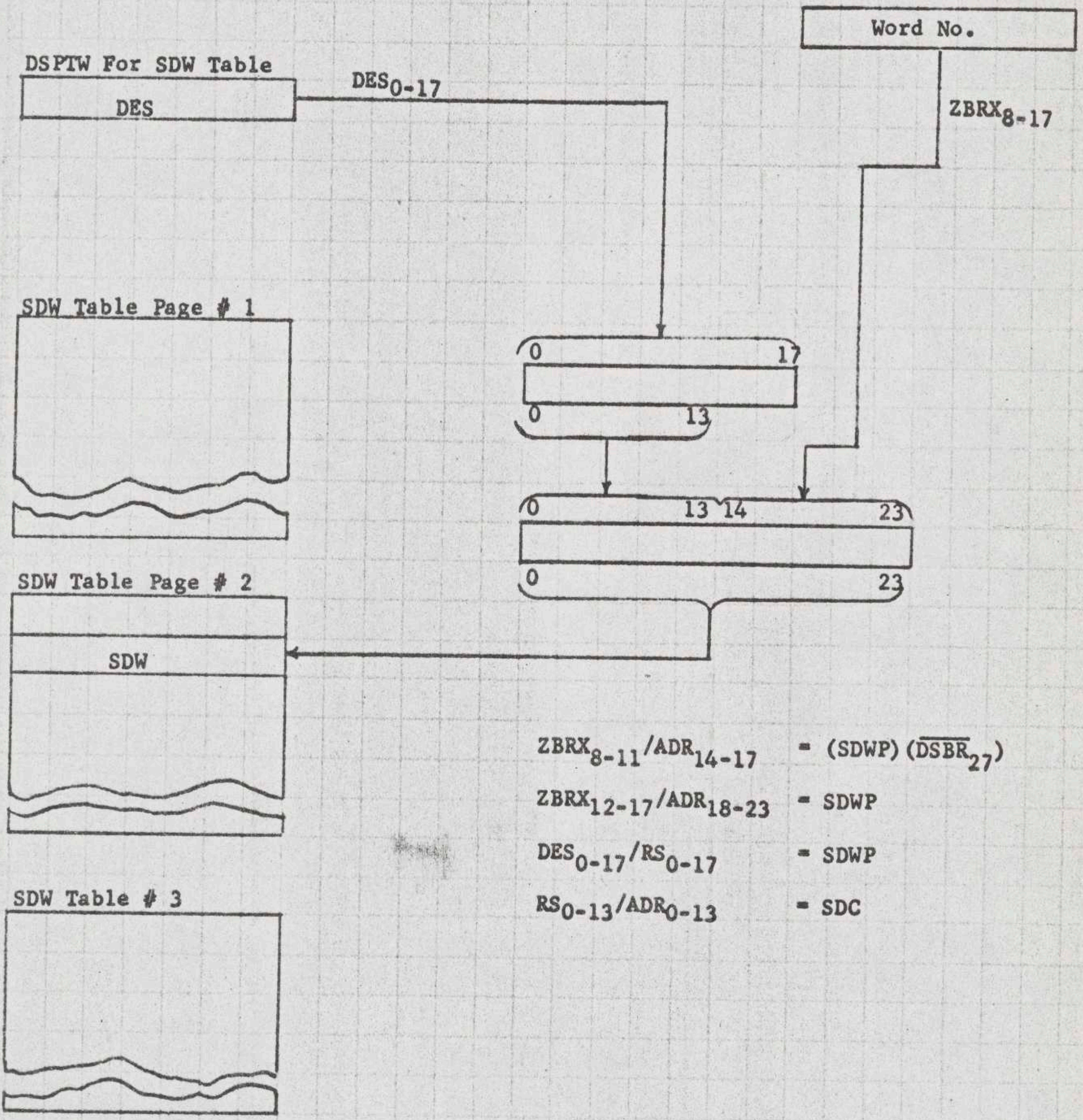
$DSBR_{0-17}/RS_{0-17}$ = DSPTW

$ZBRX_{4-11}/RS_{16-17}$, ADR_{18-23} = (DSPTW) ($DSBR_{27}$)

RS_{0-13}/ADR_{0-13} = SDC

RS_{14-17}/ADR_{14-17} = (SDC) ($\overline{SDWP}$)

Fig. 10.1.4e Descriptor Segment Page Table Word Fetch - Small Page, Flow Chart


Word No.
DSPTW For SDW Table
DES
DES_{0-17}
$ZBRX_{8-17}$
SDW Table Page # 1
0
17
0
13
0
13 14
23
0
23
SDW Table Page # 2
SDW
SDW Table # 3

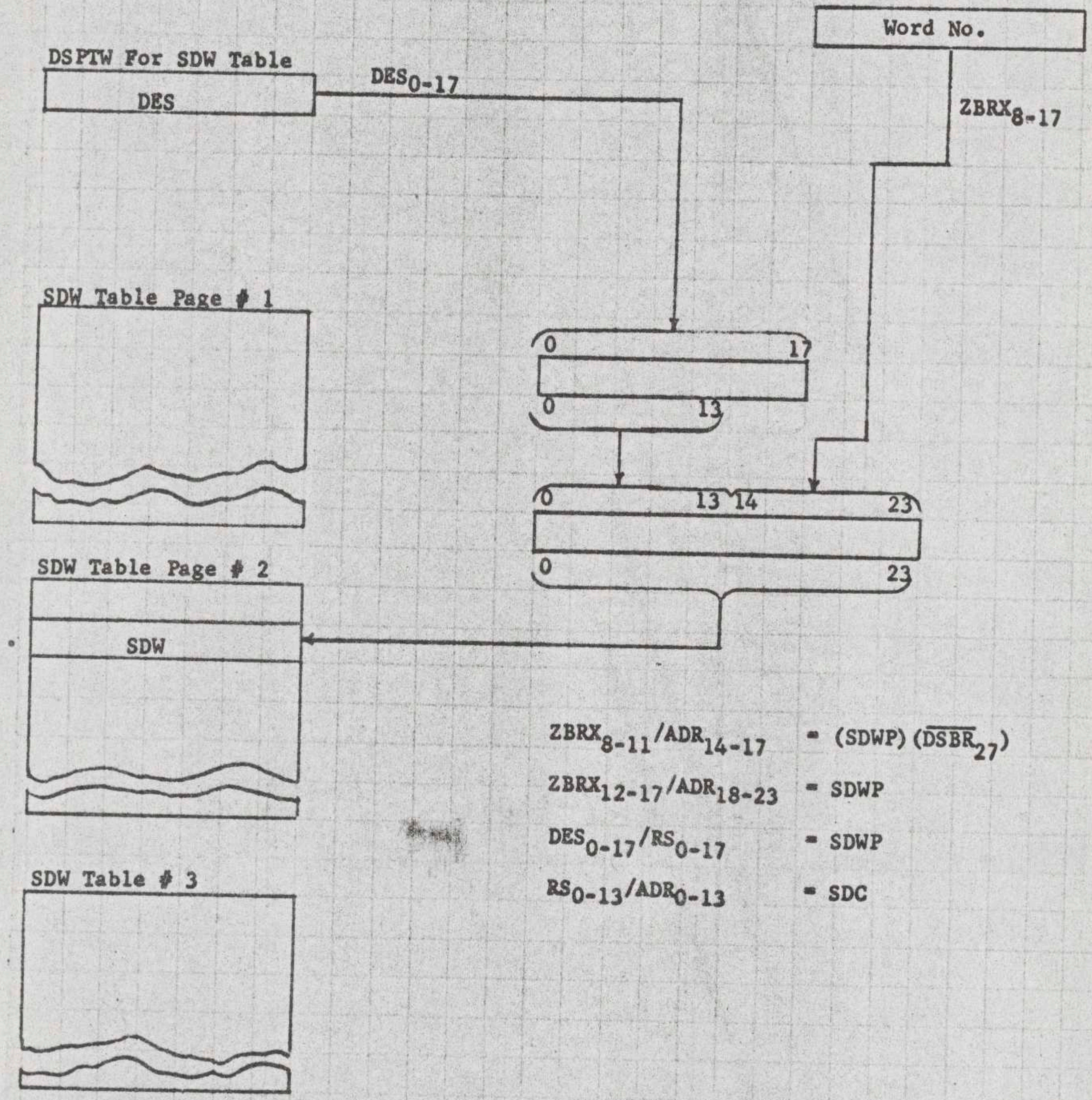
$ZBRX_{8-11}/ADR_{14-17}$ = $(SDWP)(\overline{DSBR}_{27})$

$ZBRX_{12-17}/ADR_{18-23}$ = SDWP

DES_{0-17}/RS_{0-17} = SDWP

RS_{0-13}/ADR_{0-13} = SDC

Fig. 10.1.4f SDW Fetch, Paged SWD Table (Large), Flow Chart


Word No.
DSPTW For SDW Table
DES
DES0-17
ZBRX8-17
SDW Table Page # 1
0
17
0
13
0
13 14
23
0
23
SDW Table Page # 2
SDW
SDW Table # 3

$ZBRX_{8-11}/ADR_{14-17}$ = $(SDWP)(\overline{DSBR_{27}})$

$ZBRX_{12-17}/ADR_{18-23}$ = SDWP

DES_{0-17}/RS_{0-17} = SDWP

RS_{0-13}/ADR_{0-13} = SDC

Fig. 10.1.4f SDW Fetch, Paged SWD Table (Large), Flow Chart

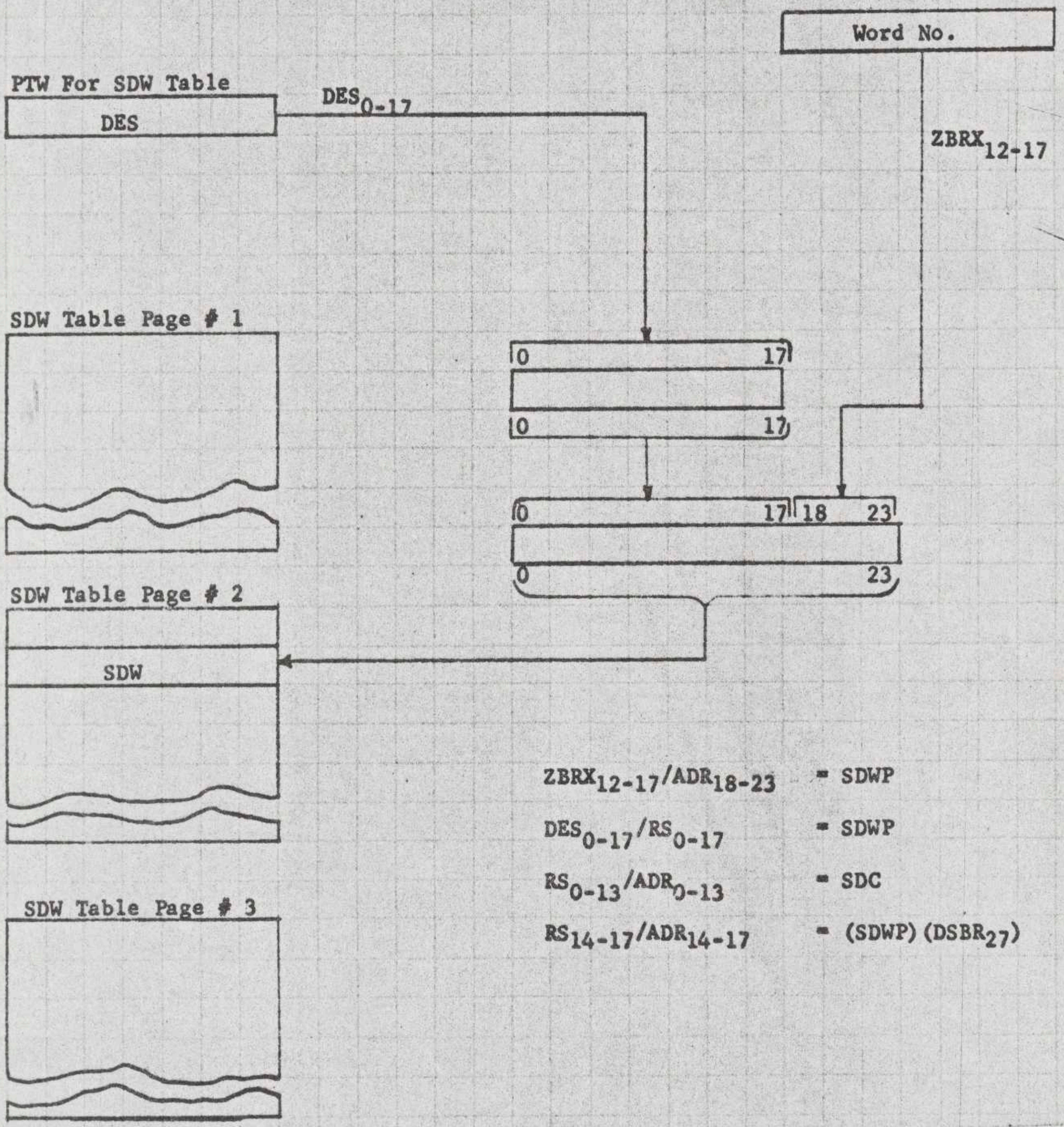
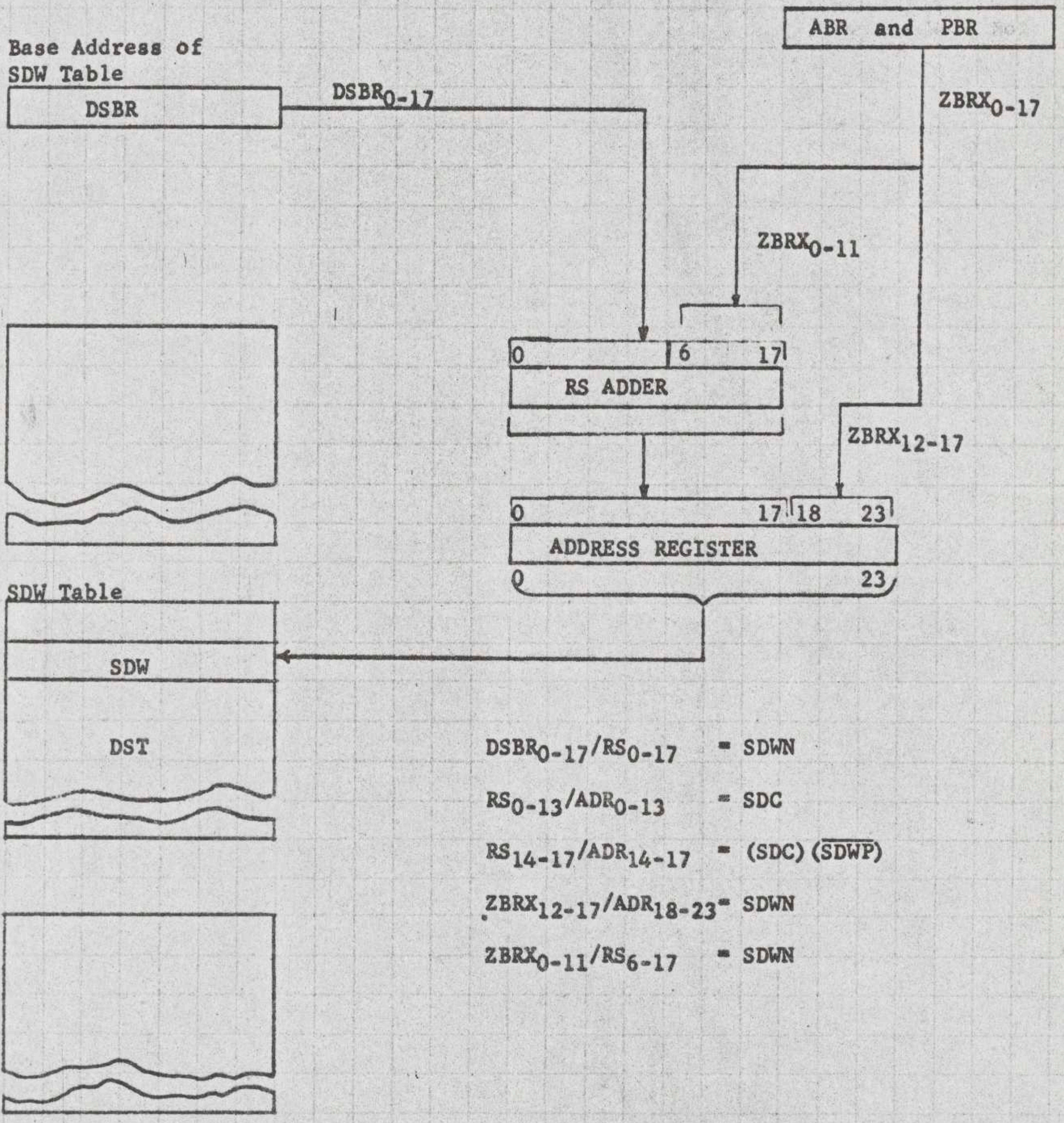

Word No.
PTW For SDW Table
DES
DES0-17
ZBRX12-17
SDW Table Page # 1
0
17
0
17
0
17
18
23
0
23
SDW Table Page # 2
SDW
SDW Table Page # 3
ZBRX12-17/ADR18-23 = SDWP
DES0-17/RS0-17 = SDWP
RS0-13/ADR0-13 = SDC
RS14-17/ADR14-17 = (SDWP)(DSBR27)

Fig. 10.1.4g SDW Fetch, Paged SWD Table (Small), Flow Chart

Cont. on Sh.10-56


ABR and PBR
Base Address of SDW Table
DSBR
DSBR0-17
ZBRX0-17
ZBRX0-11
0
6
17
RS ADDER
ZBRX12-17
0
17
18
23
ADDRESS REGISTER
0
23
SDW Table
SDW
DST

$DSBR_{0-17}/RS_{0-17}$ = SDWN

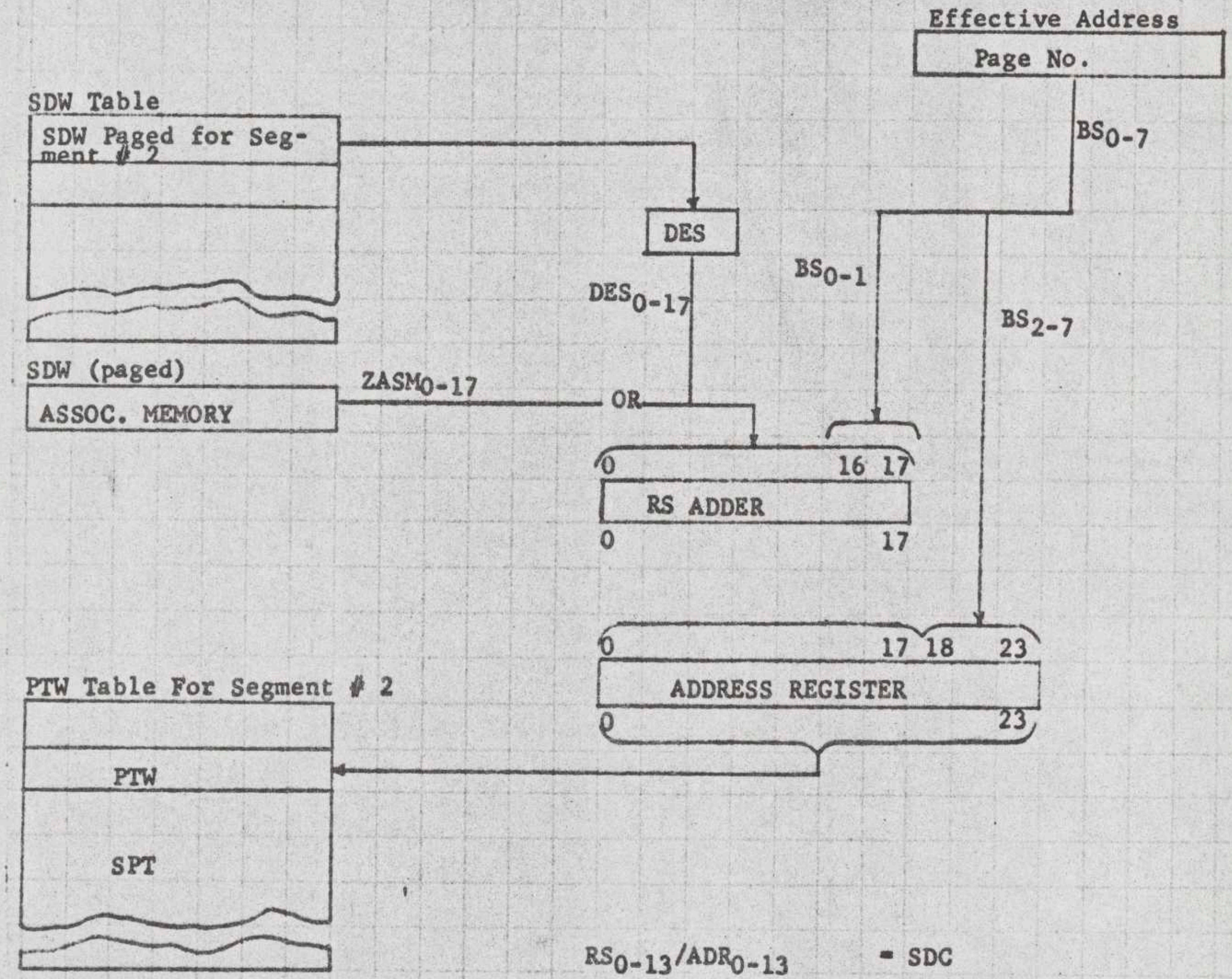
RS_{0-13}/ADR_{0-13} = SDC

RS_{14-17}/ADR_{14-17} = $(SDC)(\overline{SDWP})$

$ZBRX_{12-17}/ADR_{18-23}$ = SDWN

$ZBRX_{0-11}/RS_{6-17}$ = SDWN

Fig. 10.1.4h SDW Fetch, Nonpaged, Descriptor Segment Flow Chart


Effective Address
Page No.
SDW Table
SDW Paged for Segment # 2
BS_{0-7}
DES
BS_{0-1}
DES_{0-17}
BS_{2-7}
SDW (paged)
$ZASM_{0-17}$
OR
ASSOC. MEMORY
0
16 17
RS ADDER
0
17
0
17 18 23
ADDRESS REGISTER
0
23
PTW Table For Segment # 2
PTW
SPT

$RS_{0-13}/ADR_{0-13} = SDC$

$RS_{14-17}/ADR_{14-17} = (\overline{SDWP})(SDC)$

$BS_{0-7}/ADR_{18-23}, RS_{16-17} = (PTW)\left[(\overline{AMS})(LPF) + (AMS)(\overline{ZADS}_{27})\right]$

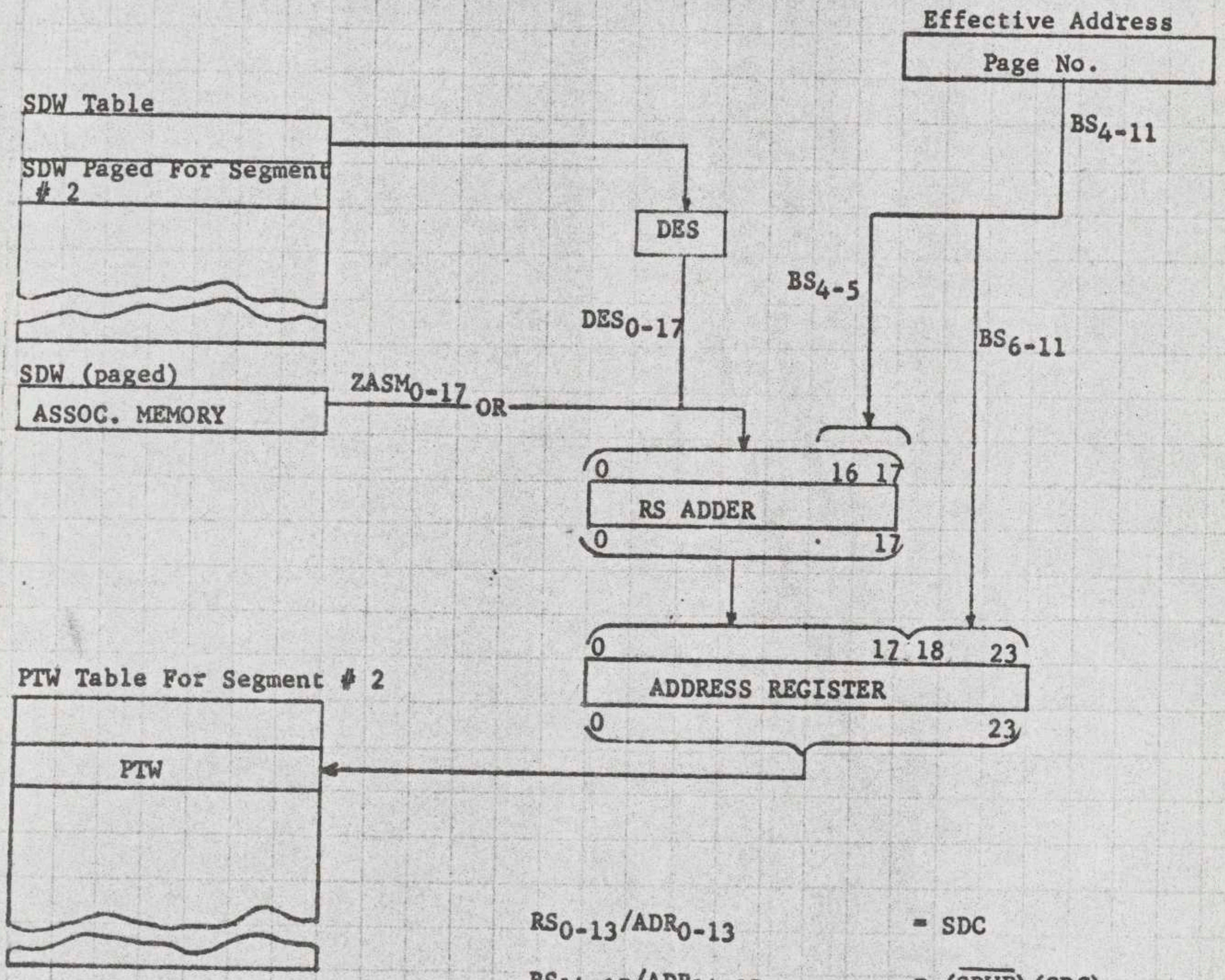
AND

$DES_{0-17}/RS_{0-17} = (\overline{AMS})(PTW)$

OR

$ZASM_{0-17}/RS_{0-17} = (AMS)(PTW)$

Fig. 10.1.4i Page Table Word Fetch - Large Page, Flow Chart


Effective Address
Page No.
BS$_{4-11}$
SDW Table
SDW Paged For Segment # 2
DES
BS$_{4-5}$
DES$_{0-17}$
BS$_{6-11}$
SDW (paged)
ASSOC. MEMORY
ZASM$_{0-17}$ OR
0
16 17
RS ADDER
0
17
0
17 18
23
ADDRESS REGISTER
0
23
PTW Table For Segment # 2
PTW

RS_{0-13}/ADR_{0-13} = SDC

RS_{14-17}/ADR_{14-17} = $(\overline{SDWP})(SDC)$

$BS_{4-11}/RS_{16-17}, ADR_{18-23}$ = $(PTW)\ [(LPF)(\overline{AMS}) + (\overline{ZADS_{27}})(AMS)]$

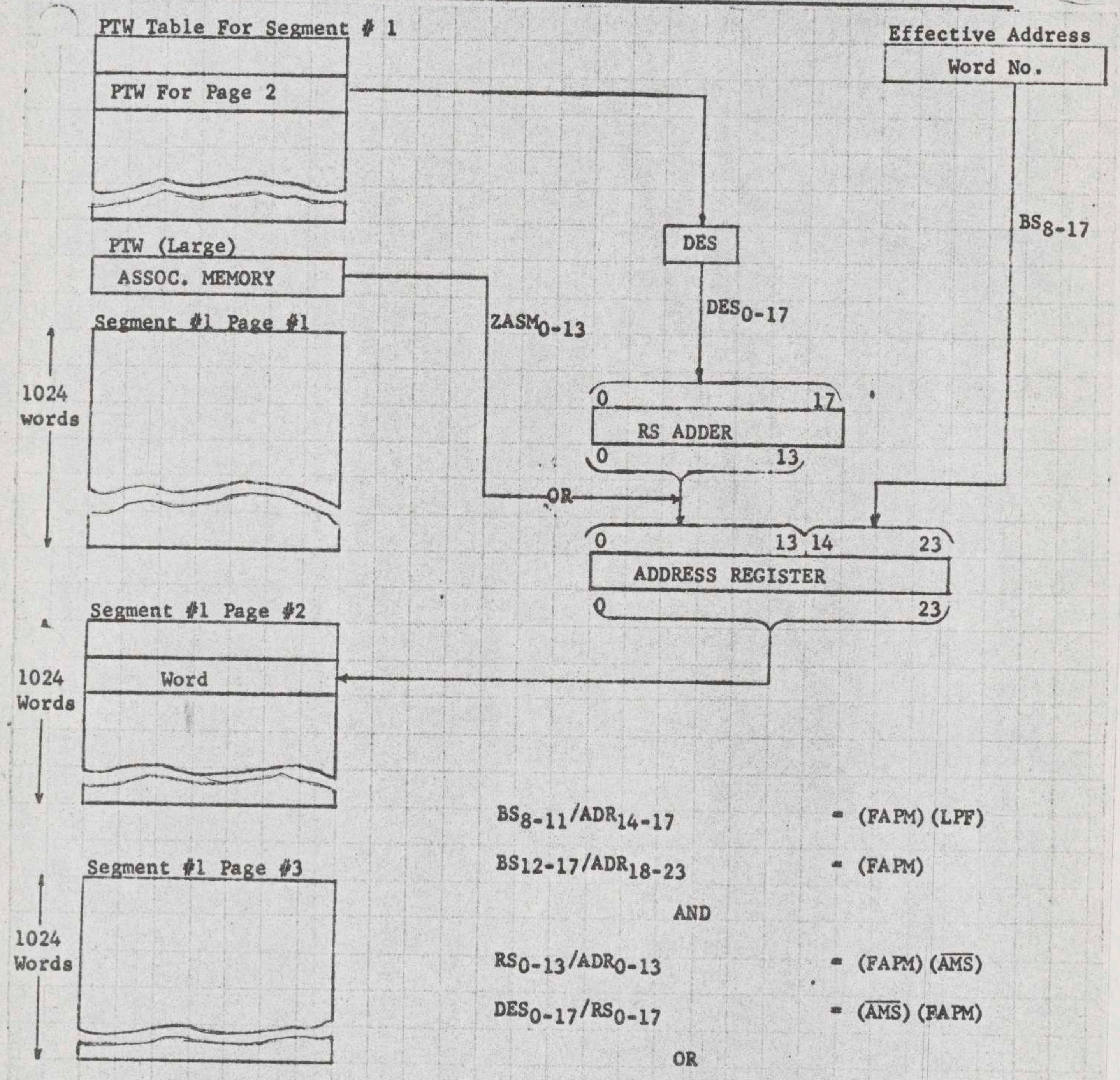
AND

DES_{0-17}/RS_{0-17} = $(\overline{AMS})(PTW)$

OR

$ZASM_{0-17}/RS_{0-17}$ = $(AMS)(PTW)$

Fig. 10.1.4j Page Table Word Fetch - Small Page, Flow Chart


PTW Table For Segment # 1
PTW For Page 2
Effective Address
Word No.
DES
BS_{8-17}
PTW (Large)
ASSOC. MEMORY
$ZASM_{0-13}$
DES_{0-17}
Segment #1 Page #1
1024 words
0
17
RS ADDER
0
13
OR
0
13
14
23
ADDRESS REGISTER
0
23
Segment #1 Page #2
1024 Words
Word
Segment #1 Page #3
1024 Words

BS_{8-11}/ADR_{14-17} = (FAPM) (LPF)

BS_{12-17}/ADR_{18-23} = (FAPM)

AND

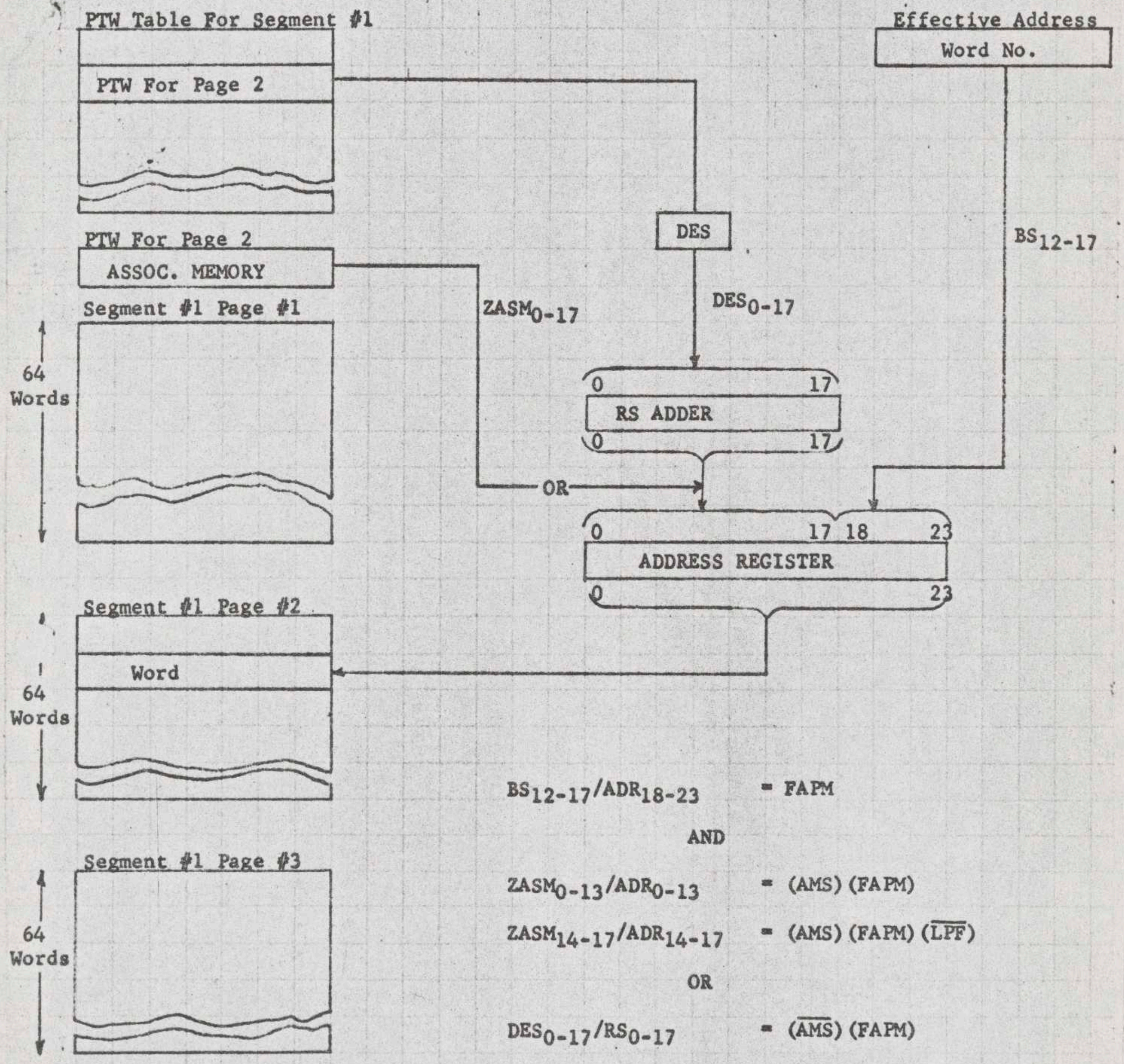
RS_{0-13}/ADR_{0-13} = (FAPM) ($\overline{AMS}$)

DES_{0-17}/RS_{0-17} = ($\overline{AMS}$) (FAPM)

OR

$ZASM_{0-13}/ADR_{0-13}$ = (AMS) (FAPM)

Fig. 10.1.4k Final Address Fetch Paged - Large Page, Flow Chart


PTW Table For Segment #1
PTW For Page 2
Effective Address
Word No.
DES
BS_{12-17}
PTW For Page 2
ASSOC. MEMORY
$ZASM_{0-17}$
DES_{0-17}
Segment #1 Page #1
64 Words
0
17
RS ADDER
0
17
OR
0
17 18
23
ADDRESS REGISTER
0
23
Segment #1 Page #2
Word
64 Words
Segment #1 Page #3
64 Words

BS_{12-17}/ADR_{18-23} = FAPM

AND

$ZASM_{0-13}/ADR_{0-13}$ = (AMS) (FAPM)

$ZASM_{14-17}/ADR_{14-17}$ = (AMS) (FAPM) $(\overline{LPF})$

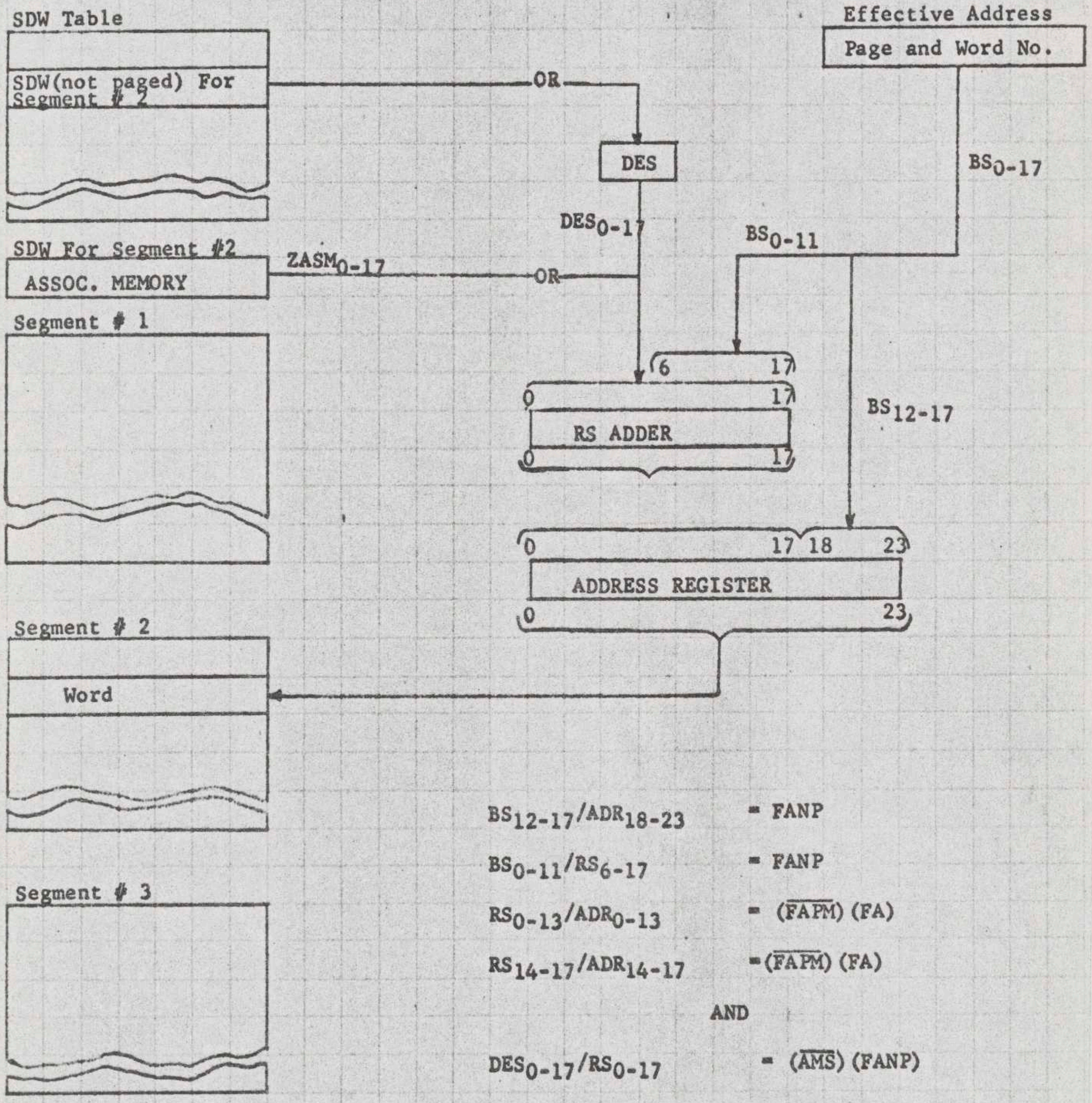
OR

DES_{0-17}/RS_{0-17} = $(\overline{AMS})$ (FAPM)

RS_{0-13}/ADR_{0-13} = (FAPM) $(\overline{AMS})$

RS_{14-17}/ADR_{14-17} = (FAPM) $(\overline{LPF})$ $(\overline{AMS})$

Fig. 10.1.4m Final Address Fetch Paged - Small Page, Flow Chart


SDW Table
SDW(not paged) For Segment # 2
OR
DES
DES0-17
SDW For Segment #2
ASSOC. MEMORY
ZASM0-17
OR
Effective Address
Page and Word No.
BS0-17
BS0-11
BS12-17
6
17
0
17
RS ADDER
0
17
0
17 18
23
ADDRESS REGISTER
0
23
Segment # 1
Segment # 2
Word
Segment # 3

BS_{12-17}/ADR_{18-23} = FANP

BS_{0-11}/RS_{6-17} = FANP

RS_{0-13}/ADR_{0-13} = $(\overline{FAPM})(FA)$

RS_{14-17}/ADR_{14-17} = $(\overline{FAPM})(FA)$

AND

DES_{0-17}/RS_{0-17} = $(\overline{AMS})(FANP)$

OR

$ZASM_{0-17}/RS_{0-17}$ = (AMS)(FANP)

Fig. 10.1.4n Final Address Fetch - Nonpaged, Flow Chart

10.1.4.2 Descriptor Generator

The Descriptor Generator receives the Descriptor field of all words from core memory, that are to be stored in the Descriptor Register. It checks for Directed Faults, Illegal Class Faults, and Access Violations as required. It generates the descriptor fields for PTWs. It presents the descriptor field to the inputs of DES_{27-35}.

Directed Fault--Anytime the Interface Frame is doing any cycle other than ABS, SWU, FAPM, or FANP ZDI lines 33-35 are checked to see if all three lines are "0." If they are the DFC line will come true indicating a Directed Fault.

Illegal Class Fault--During any cycle other than ABS, DSPTW, SWU, FAPM, or FANP the code on the three ZDI lines (33-35) will be checked. Five of these codes 000 through 100 are legitimate. If a code of 101, 110, or 111 is detected a $ICF will be sent to the Control Frame indicating an Illegal Class Fault.

Access Violation--During a FAPM or FANP cycle the descriptor field is compared with the type of operation in progress. There are four classes of faults each dealing with a particular type of Segment or Page.

1) FC1 -- The descriptor has been decoded as indicating a Data Only Segment and the current operation is:

    a) A transfer.

    b) A store against the write permit (Slave mode and bit 30 a "0").

    c) An access in violation of access rights (Slave mode and bit 31 a "0").

    d) An access during a PI cycle (Instruction Fetch).

10.1.4.2 - contd

2) FC2 -- The descriptor has been decoded as indicating a Slave Procedure Segment and the current operation is:

   a) A store against the write permit (Slave mode and bit 30 a "0").

   b) An access for an operand or indirect word in violation of access rights (Slave mode and bit 31 a "0").

3) FC3 -- The descriptor has been decoded as indicating a Procedure Only Segment and the current operation is:

   a) A store against the write permit (Slave mode and bit 30 a "0").

   b) An access for an operand or indirect word in violation of access rights (Slave mode and bit 31 a "0").

   c) An access for an operand or indirect word from another segment.

   d) A transfer from another segment, in Slave mode, to any location in this segment other than zero.

4) FC4 -- The descriptor has been decoded as indicating a Master Procedure Segment and the current operation is:

   a) An access for an operand or indirect word in Slave mode.

10.1.4.2 - contd

PTW descriptor generation - When a PTW is received from core memory bits 33-35 of its descriptor field are compared against bits 33-35 of the descriptor field of the SDW. The resultant descriptor will be as shown in Table 10.1.4.2. Bits 27 and 28 will be taken from the SDW. Bit 29 will be a 1 if bit 26 of the PTW was a "1" or if WBF is set. Bits 30-32 will be taken from the PTW if there is no Direct Fault.

Table 10.1.4.2 Resultant Descriptors for PTW-SDW Comparisons.

ZDI Lines	Descriptor Bits 33 34 35 of SDW			
33 34 35	33 34 35 0 0 1	33 34 35 0 1 0	33 34 35 0 1 1	33 34 35 1 0 0
0 0 1	0 0 1	0 0 1	0 0 1	0 0 1
0 1 0	0 0 1	0 1 0	0 1 0	0 1 0
0 1 1	0 0 1	0 1 0	0 1 1	0 1 1
1 0 0	0 0 1	0 1 0	0 1 1	1 0 0

10.1.4.3 Base Instruction, Multiple Load Store

During LREG and SREG instructions the Control Frame goes through its P cycles for each pair of operands. The Interface Frame will go through an appending cycle for each pair of operands. However, the address from the BS Adder will only be utilized once. From then on the address in the Address Register will be incremented by the INA logic. The $ADR for the first Final Address cycle will set MLSF which prevents ADR_{0-17} from being strobed and allows the increment logic to go into action. Each succeeding $ADR will strobe a new count from INA into bits 18-23 of ADR. The fourth $ADR will reset MLSF.

If the multiple load store involves a Base Frame Register BIF will be set at the same time as MLSF. If it is a store operation $ADR will be generated by a special last pulse detector before $INT. In all Base Frame multiple operations the CF will do only one P cycle and the Interface Frame will do only one appending cycle. CFR and IFR will remain set throughout the operation. MLSF and BIF control the increment logic and BIF determines whether the inputs or outputs of the Address Register will be used by the interface control. Both BIF and MLSF are reset by the last $ADR as determined by the count in ADR_{18-23}. During load Base Frame Register operations a $BORY will be sent to the Base Frame when the operand is ready. BIF is set for all Base Frame Operations except single loads.

10.1.5 MEMORY CONTROL

The Interface Frame controls all communication with the Memory Controller(s). It accepts instructions and addresses prepared by the Control Frame and by its own logic, and causes the appropriate action in the Memory Controller. It must decide which Memory Controller is to be used. It must respond to all signals received from a Memory Controller and remember which one it came from. This discussion will as much as possible deal only with the control logic. Only where necessary for clarity will functions of the address preparation logic or Memory Controller logic be discussed.

10.1.5.1 Configuration and Interlace

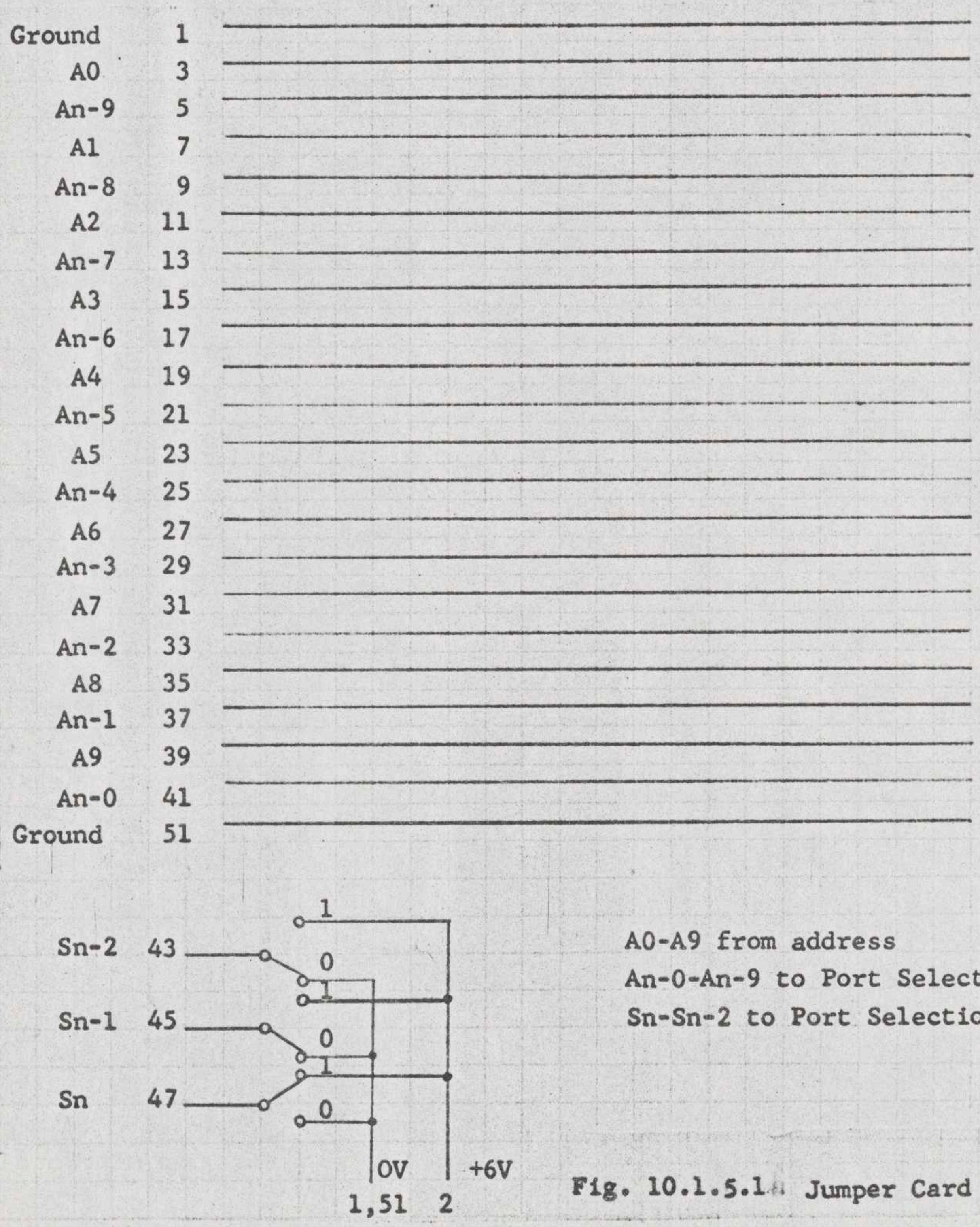
Memories of various sizes can be connected to the eight ports. The memories connected to different ports do not have to be the same size. This may result in holes in the address spectrum. The smallest size memory must be assigned the low order portion of the address spectrum. If interlace is used the memories involved must be of the same size.

The size and portion of the address spectrum assigned to each port is determined by a jumper card. There is one jumper card for each port that is used. Table 10.1.5.1 is the jumper schedule for all contemplated memory sizes. Figure 10.1.5.1 is a layout of the jumper card. Each jumper card also has three toggle switches that determines the block of addresses assigned to that port. These two factors plus the interlace control determine which port will receive the $INT. Figure 10.1.5.1a is a logic diagram of this circuitry.

Table 10.1.5.1 Jumper Schedule For Various Memory Sizes

Memory Size	A 5	B 9	C 13	D 17	E 21	F 25	G 29	H 33	J 37	K 41
16K	3	7	11	15	19	23	27	31	35	39
32K	1	3	7	11	15	19	23	27	31	35
64K	1	1	3	7	11	15	19	23	27	31
128K	1	1	1	3	7	11	15	19	23	27
256K	1	1	1	1	3	7	11	15	19	23
512K	1	1	1	1	1	3	7	11	15	19
1M	1	1	1	1	1	51	3	7	11	15
2M	1	1	1	1	1	51	51	3	7	11
4M	1	1	1	1	1	51	51	51	3	7
8M	1	1	1	1	1	51	51	51	51	3
16M	1	1	1	1	1	51	51	51	51	51

-Jumper to


Ground 1
A0 3
An-9 5
A1 7
An-8 9
A2 11
An-7 13
A3 15
An-6 17
A4 19
An-5 21
A5 23
An-4 25
A6 27
An-3 29
A7 31
An-2 33
A8 35
An-1 37
A9 39
An-0 41
Ground 51
Sn-2 43
Sn-1 45
Sn 47
1
0
1
0
1
0
0V
+6V
1,51
2

A0-A9 from address
An-0-An-9 to Port Selection
Sn-Sn-2 to Port Selection

Fig. 10.1.5.1 Jumper Card Layout

10.1.5.1 - contd

Assume that a 64 k memory is connected to port n, the switch settings are Sn-2 and Sn set to "0" and Sn-1 set to "1", and all interlace control off. Ø4 and Ø2 + Ø4 will both be low. This will inhibit gates 1, 2, and 3. By comparing the jumper card with Table 10.1.5.1 for a 64 k memory and its output to the logic diagram (Fig. 10.1.5.1b) it can be seen that to enable gate 4 bits 0 through 5 of the address must be "0" and bit 6 must be a "1." This will mean that any address between 128 k and 192 k will enable the $INT to be sent to port n, if that port is turned on.

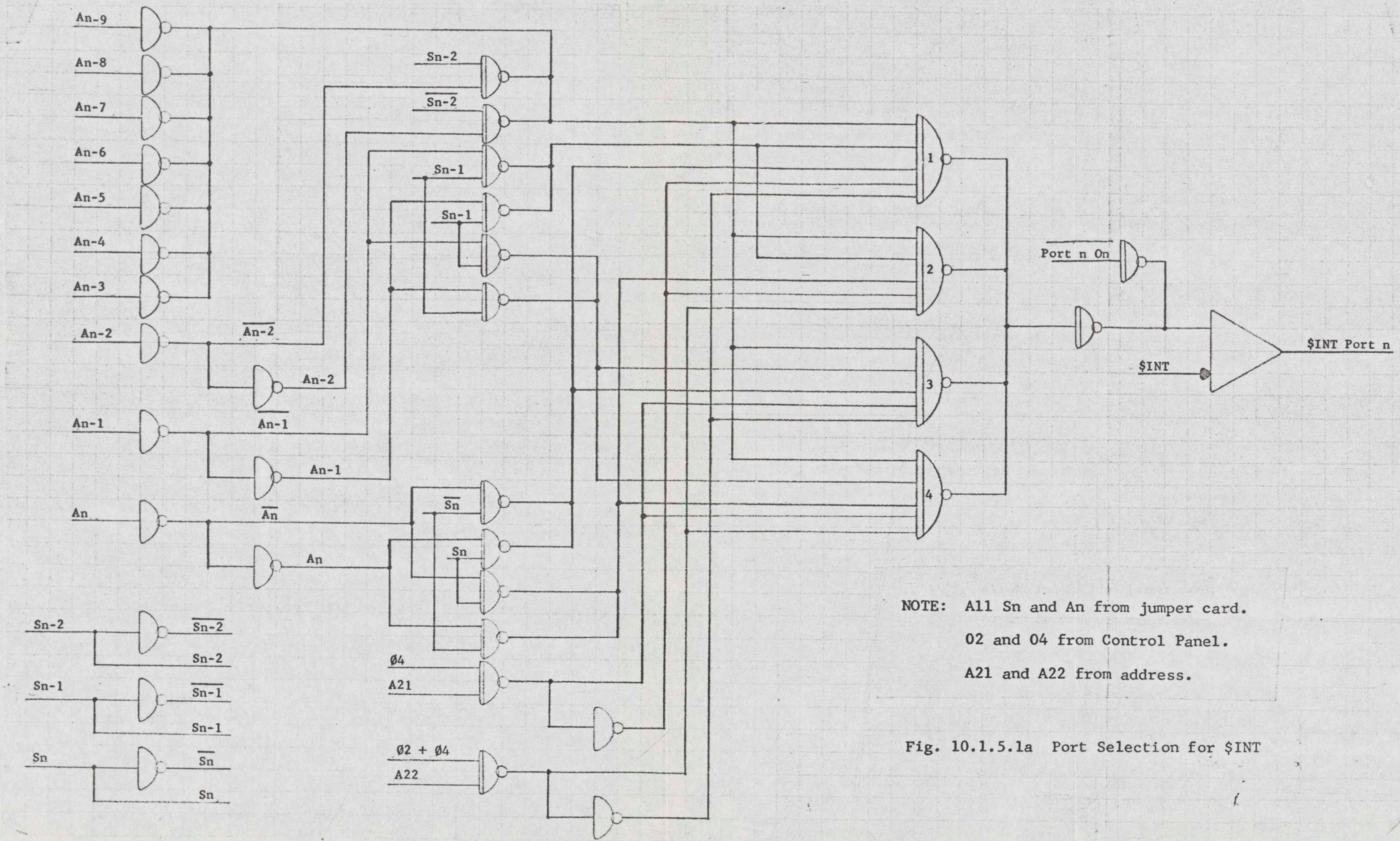
Now assume that two 32 k memories are attached to port n and port n+1. They are to be used between 0 and 64 k, and they are to be interlaced. Both jumper cards will be jumpered for 32 k as per table 10.1.5.1. Port n will have all three switches set to "0." Port n + 1 will have Sn - 2 and Sn - 1 set to "0" and Sn set to "1." In all cases address bits 0 through 7 must be "0." With Ø2 + Ø4 high the output of its gate will depend upon address bit 22. For word pair 0 and 1 bit 22 and bit 8 will be "0," which will inhibit gates 1, 2, and 3. Therefore port n will receive the $INT. For word pair 2 and 3 bit 22 will be "1" and bit 8 will be "0." Now gates 1, 2, and 4 will be inhibited. Therefore port n + 1 will receive the $INT. Each word pair up to 32 k will toggle bit 22 and alternate the port that receives the $INT. At 32 k bit 8 will become a "1" and this will reverse the order that the ports will receive the $INT. Words will be stored as per Fig. 10.1.5.1b.

10.1.5.1 - contd

Port n	Port n + 1
Address 8/9	Address 32 k + 8/9
Address 32 k + 6/7	Address 6/7
Address 4/5	Address 32 k + 4/5
Address 32 k + 2/3	Address 2/3
Address 0/1	Address 32 k + 0/1

Fig. 10.1.5.1b Word Storage For Two 32K Memories With Ø2 Interlace

Now assume that four 64 k memories are attached to port n, port n + 1, port n + 2, and port n + 3. They are to be used between 0 and 256 k, and they are to be interlaced. All four jumper cards will be jumpered for 64 k as per Table 10.1.5.1. Port n will have all switches set to "0." Port n + 1 will have Sn - 2 and Sn - 1 set to "0" and Sn set to "1." Port n + 2 will have Sn - 2 and Sn set to "0" and Sn - 1 set to "1." Port n + 3 will have Sn - 2 set to "0" and Sn - 1 and Sn set to "1." In all cases bits 0 through 5 of the address must be "0." With Ø2 + Ø4 high the output of its gate will depend upon address bit 22. With Ø4 high the output of its gate will depend upon address bit 21. For word pair 0 and 1 address bits 21, 22, 6 and 7 will be "0," which will inhibit gates 1, 2, and 3. Therefore, port n will receive the $INT. For word pair 2 and 3 address bits 21, 6, and 7 will be "0" and bit 22 will be a "1," which will inhibit gates 1, 2, and 4. Therefore, port n + 1 will receive the $INT. For word pair 4 and 5 address bits 22, 6, and 7 will be "0" and bit 21 will be a "1," which will inhibit gates 1, 3, and 4. Therefore, port n + 2 will receive the $INT. For word pair 6 and 7 address


An-9
An-8
An-7
An-6
An-5
An-4
An-3
An-2
$\overline{\text{An-2}}$
An-2
$\overline{\text{An-1}}$
An-1
An-1
An
$\overline{\text{An}}$
An
Sn-2
$\overline{\text{Sn-2}}$
Sn-1
Sn-1
$\overline{\text{Sn}}$
Sn
04
A21
02 + 04
A22
Sn-2
$\overline{\text{Sn-2}}$
Sn-2
Sn-1
$\overline{\text{Sn-1}}$
Sn-1
Sn
$\overline{\text{Sn}}$
Sn
1
2
3
4
$\overline{\text{Port n On}}$
$INT
$INT Port n

NOTE: All Sn and An from jumper card.

02 and 04 from Control Panel.

A21 and A22 from address.

Fig. 10.1.5.1a Port Selection for $INT

10.1.5.1 - contd

bits 6 and 7 will be "0" and bits 21 and 22 will be "1," which will inhibit gates 2, 3, and 4. Therefore, port n + 3 will receive the $INT. For word pair 8 and 9 the $INT will again go to port n and it will continue to rotate in this manner up to 64 k. At that point address bit 7 will become a "1" and port n + 1 will receive the $INT for word pair 64 k + 0 and 1. See Fig. 10.1.5.1c for the pattern of word storage.

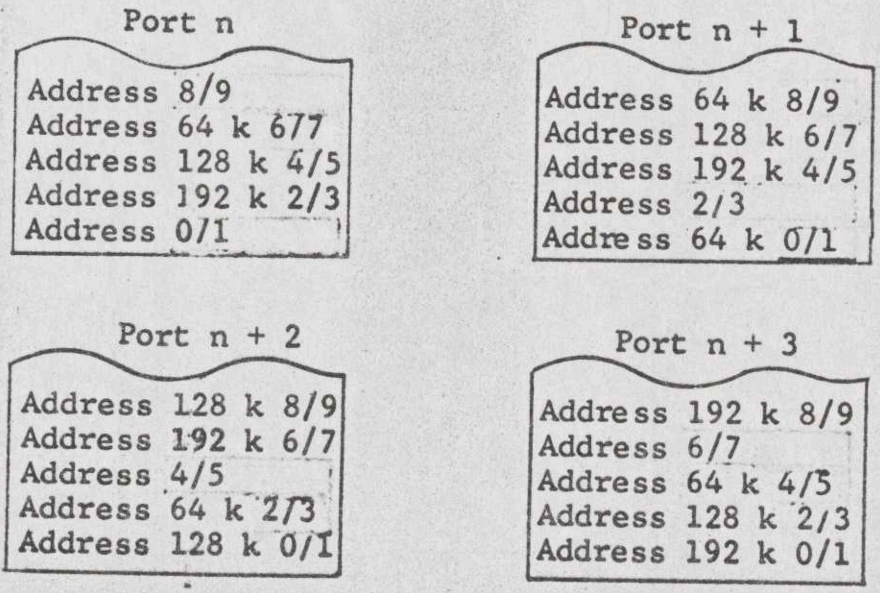

Port n
Address 8/9
Address 64 k 6/7
Address 128 k 4/5
Address 192 k 2/3
Address 0/1
Port n + 1
Address 64 k 8/9
Address 128 k 6/7
Address 192 k 4/5
Address 2/3
Address 64 k 0/1
Port n + 2
Address 128 k 8/9
Address 192 k 6/7
Address 4/5
Address 64 k 2/3
Address 128 k 0/1
Port n + 3
Address 192 k 8/9
Address 6/7
Address 64 k 4/5
Address 128 k 2/3
Address 192 k 0/1

Fig. 10.1.5.1c Word Storage For Four 64K Memories With 0 Interlace

10.1.5.2 Port Selection Flip-Flops

There are two flip-flops for each port (DIR1n and DIR2n). The $DIR1_n$ flip-flops remember what port received the last $INT. The interlace logic that selected the port also enables $INT to set the corresponding $DIR1_n$. If the operation is a Clear-Write Double-Precision or a Read-Alter-Rewrite $DIR1_n$ will select the same port to receive the $DP.

When a Memory Controller acknowledges a $INT, it will return a $ PIN. This pulse will strobe all $DIR2_n$'s. $DIR1_n$ remembers which port received the $INT, so it will control which $DIR2_n$ will set. The output of $DIR2_n$ is sent to the Base Frame to select the port to place on the ZDI lines during any read or first half of a Read-Alter-Rewrite operation. It also gates the Illegal Action Lines for the same port to the Illegal Action Register.

Any time the Control Frame recognizes an Interrupt it will go into a prepare Execute Interrupt Address cycle (PC). During this time the Interface Frame will select the highest priority port that has an Interrupt present, and enables the $INT logic and the set $DIR1_n$ for that port. When the address and the execute command for the Memory Controller are ready a $INT is issued to it at the same time that it sets $DIR1_n$. The Memory Controller responds with a $PIN, which will set $DIR2_n$ for that port, and it places the address of its highest priority Interrupt Cell that is set on the ZDI lines. The $DIR2_n$'s are decoded and used to complete the address received from the Memory Controller, which will be added to the Fault Vector switches to form the address of the instruction pair to be executed (see Fig. 10.1.5.2).

10.1.5.2 - contd

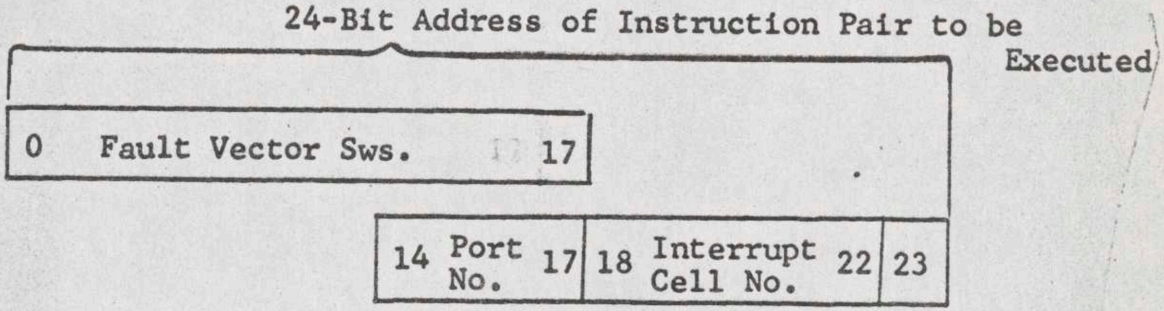


Fig. 10.1.5.2 Interrupt Address Formation

10.1.5.3 Command Logic

There are four command flip-flops that tell the Memory Controller what action is required of it. These flip-flops are strobed at the same time as the Address Register. The ZI lines from the Control Frame make up the major portion of the decode logic for these flip-flops. The operation code is decoded and the proper command flip-flops set. During appending cycles when an appending word is to be retrieved from core memory the decode logic is disabled by the lack of the FA (Final Address) level. Therefore, all four flip-flops will be reset, which is the command code for a Read-Restore Single-Precision operation. During an Indirect Word fetch EA (Effective Address) will be low preventing the ZI lines from being decoded. However, certain types of indirect sequences will cause a Read-Alter-Rewrite operation. Special conditions such as Set Written and Used bits, Execute Interrupt sequence, etc., can cause certain flip-flops to be set to perform the special operation. The output of the command flip-flops is sent to all eight ports.

10.1.5.4 Zone Lines

The zone lines control what portion of a word is changed by a Clear-Write or Read-Alter-Rewrite operation. There are eight zone flip-flops. The portion of a word that they control is as follows:

Flip-Flop	Bits Controlled
EZ00	0-5
EZU1	6-8
EZL1	9-11
EZ02	12-17
EZ03	18-23
EZU4	24-26
EZL4	27-29
EZ05	30-35

If the flip-flop is set, it will allow the corresponding bit positions to be altered. If reset, it will prevent the bits from being altered. These eight flip-flops are strobed at the same time as the address register. The decode logic that controls these flip-flops is made up of the ZI lines from the Control Frame and by the SWU level made up in the Interface Frame. During normal operation all eight flip-flops will set. During address modification the flip-flops are controlled by the op code (ZI lines 0-8) and by bits 30-35 of the instruction word (ZI lines 12-17). During instructions involving storing 18-bit operands these flip-flops are controlled by the op code (ZI lines 0-8). During a set written and used bits cycle all flip-flops except EZU4 will be reset by the SWU level.

10.1.5.5 Illegal Action Register (IAR)

If the Memory Controller operation results in an illegal action it notifies the Processor by placing a designating code on the three Illegal Action Lines and a pulse on the $IAL Line. A $IAL pulse when all three Illegal Action Lines are a "0" indicates that the operation was completed correctly.

The Interface Frame contains a 3-bit Illegal Action Register that holds the code sent by the Memory Controller. When a $IAL is received by the Interface Frame it is sent to the Base Frame and Control Frame. It also samples a gated pulse amplifier. If conditions are right it will generate a IAL_{0-2}$, which will strobe the IAR. The conditions that must be met are:

1) There has been an illegal action.

2) The Processor is not in a fault procedure.

3) There is no error code in the IAR or if there is it was placed there during a buffer filling PI.

4) The error is not a legally masked parity error.

A parity error can be legally masked unless it occurs on an appending word fetch. In that case it will be detected and it will cause a fault.

The Interface Frame will hold the error code in the IAR as long as the Control Frame needs it. When the Control Frame goes into the fault sequence the inputs to the IAR are disabled and the next $C will generate a IAL_{0-2}$, which will reset all the IAR flip-flops.

10.1.5.6 Data Out Register

All data to be stored in core memory is first stored in the Data Out Register (DOR$_{0-71}$) and held there until the memory has accepted it. During double-precision stores it will hold both words. In any case it allows the contributing frame to be released and continue about its business earlier than if that frame had to wait for memory. During fault and interrupt operation a "snapshot" is taken of various registers and flip-flops and this information is stored in DOR.

During normal store operations the frame that has data to be stored places that data on its Data Out lines. The Interface Frame proceeds with the preparation of the address. When the final effective address has been formed a $INT will be generated. This pulse will strobe the data into DOR$_{0-35}$ and will issue a $DA/S to the contributing frame. If it is a double-precision store operation the second data word will be placed on the Data Out lines and a $DP issued to the Interface Frame. This pulse strobes this data into DOR$_{36-71}$, is issued to the contributing frame as the second $DA/S, and is sent to the Memory Controller as $DP.

During PR Control Frame cycles, the tally that is to be rewritten in core memory is stored in DOR$_{54-71}$.

The output of the DOR is gated onto the Data Out Lines to memory by the DDAF flip-flop. DDAF is set during the second half of a Clear-Write Double-Precision operation or during the rewrite portion of a Read-Alter-Rewrite operation, and will gate DOR$_{36-71}$, onto the Data Out lines to memory. At all other times DDAF will be reset, and it will gate DOR$_{0-35}$ onto the Data Out lines to memory.

To be supplied later

7 To be supplied later

10.2 BASE FRAME DETAILED OPERATION

10.2.1 GENERAL DESCRIPTION

The purpose of the Base Frame is to hold address pointers to specific segments in memory, and to send these pointers to the Interface Frame and/or the Control Unit when requested by the Control Unit. To fulfill this function requires 11 basic registers: Eight Address Base Registers, a Procedure Base Register, a Temporary Base Function, and a Descriptor Segment Base Register. In addition to the 11 basic registers and the control logic required to select them, the Base Frame contains logic to enable the adding, loading, and storing of the 11 basic registers when GE-645 instructions that effect the Base Frame are decoded and the Interface Frame issues a start signal. Performing these two major functions, that is, Address Appending and Base Frame operations, requires circuits for: control point decode; access discrimination for access, Master/Slave and locked base restrictions; load base register fault detection; register selection and Base Frame counting of cycles for all multiple load/store operations.

Additional functions that the Base Frame performs which are not part of the address appending process are: Processor power on/off sequencing and fault detection and servicing for the Processor. These functions are located in the Base Frame because of space limitations within other areas of the Processor. The discussions of the additional functions are presented separately from that of address appending.

10.2.1.1 Interface Line Summary

The Interface between the Base Frame and other frames within the Processor is listed by individual line in Tables 10.2.1.1 and 10.2.1.1a. Listing is by EDA mnemonic with the last character of the mnemonic indicating the origin of the signal. Interfaces between all frames within the Processor are contained in the Processor Interface Document, G.E. drawing 43A164138.

TABLE 10.2.1.1 Base Frame Inputs

Mnemonic	Description
	Register Frame
QOUGD	\$GS.$\overline{GF}$
	Control Frame
BBCRB(BCR)	INSTRUCTION IS REPEAT OR STORE
BCDBB(CDB0-17/)	CONTROL FOR DB-REG./ZDIBF0-17
BCTAB(CT)	TAG REG.MODIFICATION IDC OR DIC
BDDBB(CDB0-17/)	CONTROL FOR DB-REG./ZDIRF18-35)
DBPIB(SPI)	DECODE(SPI(DPI AND EXI AND RPI
BEADB(EA)	PREPARED ADDRESS IS FOR OPERAND
BHLTB(HALT)	LOGIC DECODE FOR STOP
BICFB(CSETIC)	LEVEL TO SET IC FLIP FLOPS
BICRB(CRESET IC)	LEVEL TO RESET IC FLIP FLOPS
BIPCB(PC)	INTERRUPT PRIORITY CHANNEL
BIPFB(IPF)	DECODE LOGIC IPR STROBE
BMPAB(FPA)	FAULT OR EXECUTE INT. IN PA CYCLE
BPNLB(CPNL SW/ZDI)	CONTROL FOR PANEL SWTCHS/ZDIRF0(35
BSPAB(CSET PA)	LEVEL TO SET PA FLIP FLOP
BSPIB(CSET PI)	LEVEL TO SET PI FLIP FLOP
BSPNB(CSET PN)	LEVEL TO SET PN FLIP FLOP
BSPTB(CSET PT)	LEVEL TO SET PT FLIP FLOP
BSPZB(CSET PZ)	LEVEL TO SET PZ FLIP FLOP
BSWNB(CSET WN)	LEVEL TO SET WN FLIP FLOP
BTDLB(TMCH(2)	TOO MUCH COUNTER EQUALS 2
BTDMB(TMCH(1)	TOO MUCH COUNTER EQUALS 1
BTDNB(TMCH(0)	TOO MUCH COUNTER EQUALS 2
BTRGB(TRG01)	TRANSFER GO GROUP SUB 1 DECODE
BYCFB(Y)	DIRECT OPERAND THE Y OF IT
BZERB(CZEROES/)	CONTROL FOR ZEROES/ZDIRF0-17
DA00B-DA02B(ZY0-2)	INST. REG. SW. EVEN

TABLE 10.2.1.1 · Base Frame Inputs - Cont'd.

Mnemonic	Description
	Control Frame - Cont'd.
DA29B(ZY29)	INST. REG. SW. EVEN
DB00B-DB02B(ZC0-2)	INST. REG. SW. ODD
DB29B(ZC29)	INST. REG. SW. ODD
DW00B-DW08B(ZI0-8)	INST. REGISTER DECODE BUS
DW12B-DW17B(ZI12-17)	INSTRUCTION REGISTER DECODE BUS
ECICB(CIOC)	
EDVKB(DVCK)	DIVIDE CHECK FF
EEXFB(EXF)	EXECUTE FF
EFLFB(FOLF)	FAULT OVERFLOW FF
EKC1B(1KC)	TIMER 1KC OUTPUT
ELSTB(LSTi)	
EMSYB(M/S)	MASTER/SLAVE FLAG ICTA8
EPINB(PIN)	
ERPDB(FD)	REPEAT DOUBLE MODE
ERPTB(FT)	REPEAT MODE
ESTPB(STOP)	STOP FLIP FLOP
ETERB(TERM)	TERMINATION OF INSTRUCTION
EWIWB(WI2)	W WAIT FLAG SUB 2 FLIP FLOP
EXDEB(XDE)	EXECUTE DOUBLE FROM EVEN LOCATION
MB00B-MB17B(BS0-17)	ADDRESS BASE ADDER
QIPRB($IPR)	STROBE TO SET PIR FLIP FLOPS
QSC0B($C)	
QZCRB($CSR)	
QZSRB($R)	
	Interface Frame
BFAAM(FA)	FINAL ADDRESS FETCH [FAPM+FANP+ABS]
EIFAM-EIFCM(IARA-C)	3-BIT ILLEGAL ACTION REG.
EP20M-EP27M(DIR20-7)	
FD30M-FD32M(DFA-C)	
QAVFM($AVF)	ACCESS VIOLATION FAULT PULSE
QBORM($BORY)	START BASE OPERATION PULSE

TABLE 10.2.1.1 Base Frame Inputs - Cont'd.

Mnemonic	Description
	Control Frame - Cont'd.
QBORM($BORY)	START BASE OPERATION PULSE
QDFCM($DF)	DIRECTED FAULT PULSE
QDSTM($DS)	DATA STORED PULSE
QIASM($SIAL)	SPECIAL ILLEGAL ACTION PULSE
QICFM($ICF)	ILLEGAL CLASS FAULT PULSE
QINTM($INT)	GENERAL STROBE
QOBFM($OBF)	OUT OF BOUNDS FAULT PULSE
QOCNM($CON)	CONNECT PULSE FROM ALL PORTS
QOIAM($IAL)	ILLEGAL ACTION PULSE FROM ALL PORTS
QPINM($PIN)	DELAYED QOPNM PULSE
QRSFM($RSF)	IF READY TO FAULT PULSE
QSDAM($DA)	READ ONLY PULSE
	Maintenance Panel
SD00T-SD35T	DATA SWITCHES
SIHFT	INHIBIT FAULT SWITCH
SIHFT	INHIBIT FAULT SWITCH
SL00T-SL17T	FAULT VECTOR SWITCHES
SLPAT-SLPHT	PORT SWITCHES A-H
SPOFT	POWER OFF SWITCH
SPONT	POWER ON SWITCH
SQB1T	TEST CLOCK SWITCH
SQB1T	TEST CLOCK SWITCH
SSTFT	STOP ON FAULT SWITCH
SSTFT	STOP ON FAULT SWITCH
STSTT	TEST SWITCH
STSTT	TEST SWITCH

TABLE 10.2.1.1a Base Frame Outputs

Mnemonic	Description
	Register Frame
DV00F-DV35F (ZDI RF0-35)	DATA INPUT TO REGISTER FRAME
	Control Frame
BCLTF($\overline{\text{FLT}}$)	RECOGNIZE FAULT TAG 1-3
BEOTF(BIT29?ITS?ITB)	
BFRQF(FRQ)	GROUP 6 FAULT PRESENT
BONCF(GP2 PONC)	GP2 AND PONC
BPETF(PBR=TBR)	CONTENTS OF PBR EQUALS TBR CONTENTS
BRR0F($\overline{\text{RR}}$)	RELEASE $R
BS03F(RS3)	RESET GROUP 3.4.5.6 FAULTS
BSBTF(SABT)	START ABORT SEQUENCE
BSFFF($\overline{\text{SFF}}$)	STOP FOR FAULTS
DFA0F-DFA4F(FAA-E)	FAULT ADDRESS ENCODES FA0 MOST SIG.
DI00F-DI35F(ZDI0-35)	DATA INPUT BUS
DOMRF	MEMORY READY
DZ00F-DZ17F(ZBRI0-17)	BASE REGISTER INTERNAL BUS
EDPIF(DPI)	DOUBLE PI CYCLE
EGB0F(GB0)	GENERAL BASE INSTRUCTION F/F
EINAF	INITIALIZE
EN03F-EN05F(INC3-5)	INCREMENTOR REGISTER
EPE0F($\overline{\text{PEO}}$)	PARITY ERROR IN OPERAND
EPFIF(PFPI)	PARITY ERROR IN BUFFRD. INSTRCN.
EPWPF(PWPI)	PRTY.ERR.IN PTW OR SDW FOR BUFFRD. INSTRCN.
ERSTF	REMOTE STOP F/F
ESEPF(SEP)	PRTY.ERR.IN OU OPERAND
ESN0F(SN0)	SNAPSHOT ZERO F/F
ESN1F(SN1)	SNAPSHOT ONE F/F
EWCCF($\overline{\text{WC}}$)	WAIT FOR EXECUTE INTRPT.CELL NO.CYCLE
EWF0F(WF)	ABORT F/F
EWFBF(WFTB)	WAIT FOR TROUBLE FLIP-FLOP
QABTF($ABT)	END OF WF PULSE

TABLE 10.2.1.1a Base Frame Outputs

Mnemonic	Description
	Control Frame - Cont'd.
QSN0F($SN0)	END OF SNAPSHOT 0 PULSE
QSNZF	END OF SNAPSHOT 1 PULSE
QZBCF($BC)	DATA IS READY
QZCEF($ESTR)	STROBE FOR EVEN SEGMNT TAG REGISTER
QZCOF($OSTR)	STROBE FOR ODD SEGMENT TAG REGISTER
	Associative Memory Frame
BCAMF(CAM)	CLEAR ASSOC.MEMORY
BSAMF(SAM)	STORE ASSOC.MEMORY
BSAZF(SAZ)	STORE ASSOC.ZERO
DX00F-DX17F(ZBRX0-17)	BASE REGISTER EXTERNAL BUS
EN01F-EN04F(INC1-4)	INCREMENTOR REGISTER
EWF0F(WF)	WF ABORT F/F
	Interface Frame
BAB4F	PH. 4 SIG. TO PORTS A+B INTERLACE
BABIF	PH. 2 OR 4 SIG. TO PORTS A+B INTERLACE
BCD4F	PH. 4 SIG. TO PORTS C+D INTERLACE
BCDIF	PH. 2 OR 4 SIG. TO PORTS C+D INTERLACE
BEF4F	PH. 4 SIG. TO PORTS E+F INTERLACE
BEFIF	PH. 2 OR 4 SIG. TO PORTS E+F INTERLACE
BGH4F	PH.4 SIG. TO PORTS G+H INTERLACE
BGHIF	PH. 2 OR 4 SIG. TO PORTS G+H INTERLACE
BP04F(GP4)	RECOGNIZE BUFFR.FILLING PI PARITY FAULT
BPETF(PBR=TBR)	CONTENTS OF PBR EQUALS CONTENTS OF TBR
BR00F-BR17F	REM.OR LOC.CONTRL.FAULT VECTOR SWITCHES
BRPAF-BRPHF(PORT ON)	A-H PORT ON
BSFFF(SFF)	STOP FOR FAULTS
DD00F-DD35F(ZD0BF0-35)	36-BIT BASE FRAME DATA OUT BUS
DI00F-DI35F(ZDI0-35)	DATA INPUT BUS
DX00F-DX17F(ZBRX0-17)	BASE REGISTER EXTERNAL BUS

TABLE 10.2.1.1a Base Frame Outputs - Cont'd.

Mnemonic	Description
	Interface Frame - Cont'd.
EGB0F(GB0)	GENERAL BASE INSTRUCTION F/F
EN03F-EN05F(INC3-5)	INCREMENTOR REGISTER
EPNBF(PNB)	$PIN AND BUFFRD.INSTRCN.REQUEST
ESN0F(SN0)	SNAPSHOT ZERO F/F
ESN1F(SN1)	SNAPSHOT ONE F/F
EWF0F(WF)	WF ABORT F/F
FS00F-FS28F($\overline{\text{DSBR0-28}}$)	29-BIT DESCRIPTOR SEG.BASE REGISTER
QBRYF($BRY)	BASE READY PULSE
QDPBF($DP)	
QSN0F($SN0)	SNAPSHOT ZERO PULSE
QSNZF($SN1)	SNAPSHOT ONE PULSE
	Relay Box
DXB0F-DXB2F ZXBN0-2	EXTERNAL BASE NUMBER BUS
EES0F-EES3F ESTR0-3	4-BIT EVEN SEG. TAG REGISTER
EOS0F-EOS3F OSTR0-3	4-BIT ODD SEG. TAG REGISTER
FA00F-FA23F ABR0 0-23	23-BIT ADDRESS BASE REG. NO.0
FB00F-FB23F ABR1 0-23	23-BIT ADDRESS BASE REG. NO.1
FC00F-FC23F ABR2 0-23	23-BIT ADDRESS BASE REG. NO.2
FD00F-FD23F ABR3 0-23	23-BIT ADDRESS BASE REG. NO.3
FE00F-FE23F ABR4 0-23	23-BIT ADDRESS BASE REG. NO.4
FF00F-FF23F ABR5 0-23	23-BIT ADDRESS BASE REG. NO.5
FG00F-FG23F ABR6 0-23	23-BIT ADDRESS BASE REG. NO.6
FH00F-FH23F ABR7 0-23	23-BIT ADDRESS BASE REG. NO.7
FP00F-FP17F PBR0-17	18-BIT PROCEDURE BASE REGISTER
FS00F-FS28F DSBR0-28	29-BIT DESCRIPTOR SEG. BASE REG.
FT00F-FT17F TBR0-17	18-BIT TEMPORARY BASE REGISTER

TABLE 10.2.1.1a Base Frame Outputs -contd

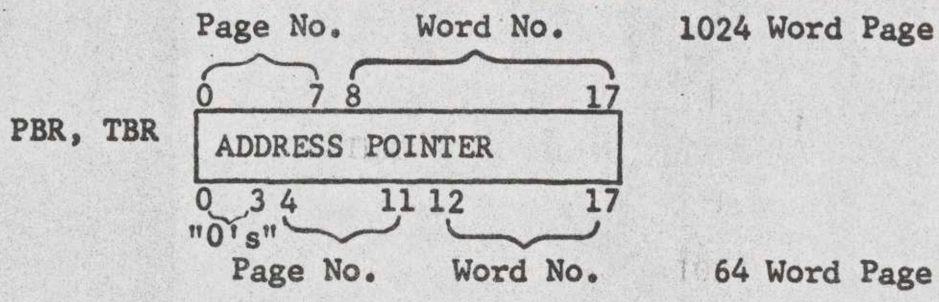
Mnemonic		Description
		Maintenance Panel
EAPDF	PAPZPNPT	APPENDING CONTROL F/F
EGBOF	GBO	GENERAL BASE INSTRUCTION F/F
EINAF		INITIALIZE POWER F/F
EITBF	ITB	INDIRECT TO BASE F/F
EITSF	ITS	INDIRECT TO SEGMENT F/F
EN00F-EN05F	INC0-5	INCREMENTOR REGISTER
ERWNF	IR/RI	WAIT FOR INDIRECT WITH PREVIOUS IR/RI -F/F
EWFMF	WFM	WAIT FOR MODIFIER F/F
		Junction Panel
JA00F-JA35F		PORT A MEMORY DATA
JAMRF		PORT A MEMORY READY
JB00F-JB35F		PORT B MEMORY DATA
JBMRF		PORT B MEMORY READY
JC00F-JC35F		PORT C MEMORY DATA
JCMRF		PORT C MEMORY READY
JD00F-JD35F		PORT D MEMORY DATA
JDMRF		PORT D MEMORY READY
JE00F-JE35F		PORT E MEMORY DATA
JEMRF		PORT E MEMORY READY
JF00F-F35F		PORT F MEMORY DATA
JFMRF		PORT F MEMORY READY
JG00F-JG35F		PORT G MEMORY DATA
JGMRF		PORT G MEMORY READY
JH00F-JH35F		PORT H MEMORY DATA
JHMRF		PORT H MEMORY READY
JRSAF		PORT A REMOTE STOP
JRSBF		PORT B REMOTE STOP
JRSCF		PORT C REMOTE STOP
JRSDF		PORT D REMOTE STOP
JRSEF		PORT E REMOTE STOP
JRSFF		PORT F REMOTE STOP
JRSGF		PORT G REMOTE STOP
JRSHF		PORT H REMOTE STOP
KPONF		SEQUENCER PWR. ON CONTRACTOR
VP25F		?25-VOLTS DC
SAB4N		INTERLACE PORTS A+B 4 PHASE SELECTION
SABIN		INTERLACE PORTS A+B 2 OR 4 PHASE SELECTION

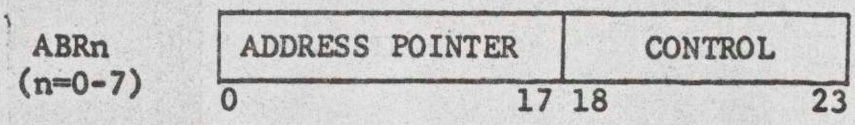
TABLE 10.12.1.1a Base Frame Outputs - contd

Mnemonic	Description
SCD4N	INTERLACE PORTS C+D 4 PHASE SELECTION
SCDIN	INTERLACE PORTS C+D 2 or 4 PHASE SELECTION
SEF4N	INTERLACE PORTS E+F 4 PHASE SELECTION
SEFIN	INTERLACE PORTS E+F 2 OR 4 PHASE SELECTION
SGH4N	INTERLACE PORTS G+H 4 PHASE SELECTION
SGHIN	INTERLACE PORTS G+H 2 OR 4 PHASE SELECTION
SROON-SRI7N	REMOTE PANEL FAULT VECTOR SWITCHES
SRPAN-SRPHN	PORT ENABLE A-H SWITCHES
SRPSN	INTERLACE

10.2.1.2 Register Formats

Prior to any discussion of Base Frame operations, it is necessary to describe the format of the 11 basic registers and the use to which specific fields within each register are put. Subsequent operations within the Base Frame are based to a large extent, on the contents of the registers involved. The various register formats are shown in Fig. 10.2.1.2.


Page No.
Word No.
1024 Word Page
0
7
8
17
PBR, TBR
ADDRESS POINTER
0
3
4
11
12
17
"0's"
Page No.
Word No.
64 Word Page


ABRn
(n=0-7)
ADDRESS POINTER
CONTROL
0
17
18
23

Bits

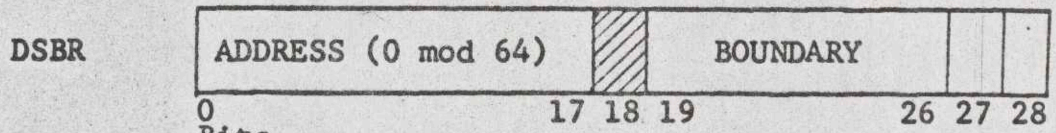
0-17 - Address pointer as in the PBR, TBR

18-20 - External base register designator (0-7)

21 - Internal/External base bit ("0"/"1" respectively)

22 - Lock base bit (0 = Master/Slave; 1 = Master only)

23 - Unassigned


DSBR
ADDRESS (0 mod 64)
BOUNDARY
0
17
18
19
26
27
28

Bits

0-17 - Base address of segment (0 mod 64)

19-26 - Boundary (number of pages if paged; number of blocks if nonpages.

27 - Page Size 64/1024 ("1"/"0" respectively)

28 - Descriptor segment paged or nonpaged ("0"/"1" respectively)

Fig. 10.2.1.2 Base Frame, Register Formats

10.2.2 BASIC OPERATING CYCLES

The basic operating cycles for address appending within the Base Frame are designated by the set condition of certain flags anagalous to the various prepare and wait flags of the Control Unit. These flags define:

1) Appending for the preparation of the address of an instruction (PI).

2) Appending for the preparation of the address of an operand (PA, PZ, PN, PT).

3) Preparing for the expected receipt of an ITS or ITB tag modified word that is pulled as the result of an RI, or IR modification (WFM).

4) Performing a series of operations that are considered general base operations because they involve the adding, loading, storing, or movement of data into or out of the 11 registers that contain address pointers and related control information (GBO).

The general base operations cycle (GBO) may occur during an appending for operands cycle if a base instruction is received by the Processor and is being processed. Additionally, all operations within the Base Frame may be ignored by the Interface Frame if the Processor is addressing the Absolute mode.

10.2.2.1 Base Frame Block Diagram

General Electric drawing 43D160212, sheet 9, shows the major data flow paths between the various elements within the Base Frame. The control points and strobes shown are by EDA and engineering mnemonic. Detailed logic diagrams for the control circuitry are included or are referenced under paragraph 10.3.

NOTE: All discussion of the Base Frame in the following paragraphs is in reference to 43D160212, sheet 9, and the simplified flow chart of Fig. 10.2.2.1.

NOTE: THE $C THAT PUTS THE BASE FRAME INTO THE WFM CYCLE IS THE $C THAT FINISHES THE PREVIOUS PA, PZ, OR PN CYCLE

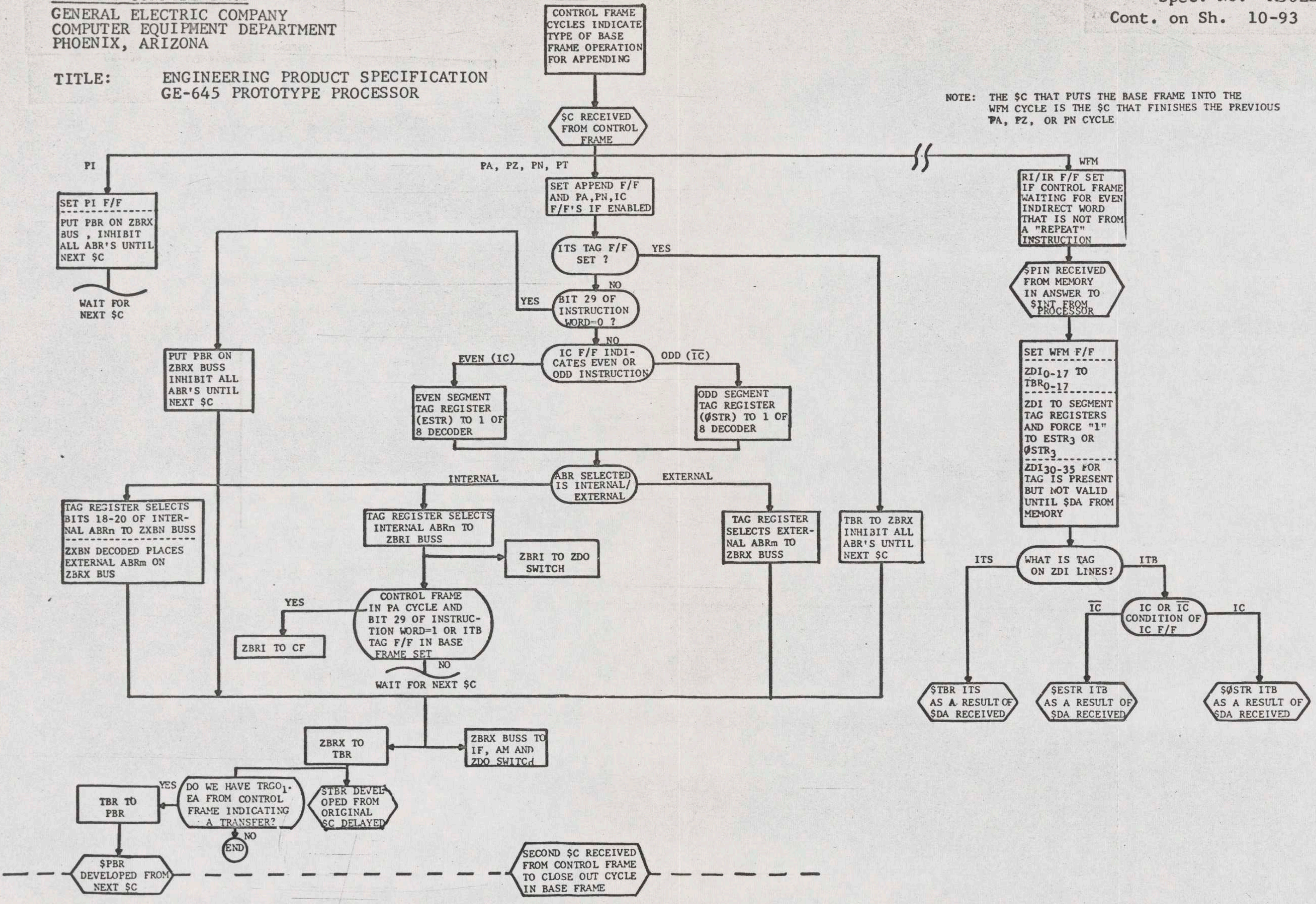

CONTROL FRAME CYCLES INDICATE TYPE OF BASE FRAME OPERATION FOR APPENDING
$C RECEIVED FROM CONTROL FRAME
PI
PA, PZ, PN, PT
WFM
SET PI F/F
PUT PBR ON ZBRX BUS , INHIBIT ALL ABR'S UNTIL NEXT $C
WAIT FOR NEXT $C
SET APPEND F/F AND PA,PN,IC F/F'S IF ENABLED
ITS TAG F/F SET ?
YES
NO
BIT 29 OF INSTRUCTION WORD=0 ?
YES
NO
PUT PBR ON ZBRX BUSS INHIBIT ALL ABR'S UNTIL NEXT $C
IC F/F INDICATES EVEN OR ODD INSTRUCTION
EVEN (IC)
ODD ($\overline{IC}$)
EVEN SEGMENT TAG REGISTER (ESTR) TO 1 OF 8 DECODER
ODD SEGMENT TAG REGISTER (ØSTR) TO 1 OF 8 DECODER
ABR SELECTED IS INTERNAL/ EXTERNAL
INTERNAL
EXTERNAL
TAG REGISTER SELECTS BITS 18-20 OF INTERNAL ABRn TO ZXBN BUSS
ZXBN DECODED PLACES EXTERNAL ABRm ON ZBRX BUS
TAG REGISTER SELECTS INTERNAL ABRn TO ZBRI BUSS
ZBRI TO ZDO SWITCH
TAG REGISTER SELECTS EXTERNAL ABRm TO ZBRX BUSS
TBR TO ZBRX INHIBIT ALL ABR'S UNTIL NEXT $C
CONTROL FRAME IN PA CYCLE AND BIT 29 OF INSTRUCTION WORD=1 OR ITB TAG F/F IN BASE FRAME SET
YES
ZBRI TO CF
NO
WAIT FOR NEXT $C
ZBRX TO TBR
ZBRX BUSS TO IF, AM AND ZDO SWITCH
DO WE HAVE TRGO$_1$·EA FROM CONTROL FRAME INDICATING A TRANSFER?
YES
NO
END
TBR TO PBR
$TBR DEVELOPED FROM ORIGINAL $C DELAYED
$PBR DEVELOPED FROM NEXT $C
SECOND $C RECEIVED FROM CONTROL FRAME TO CLOSE OUT CYCLE IN BASE FRAME
RI/IR F/F SET IF CONTROL FRAME WAITING FOR EVEN INDIRECT WORD THAT IS NOT FROM A "REPEAT" INSTRUCTION
$PIN RECEIVED FROM MEMORY IN ANSWER TO $INT FROM PROCESSOR
SET WFM F/F
ZDI$_{0-17}$ TO TBR$_{0-17}$
ZDI TO SEGMENT TAG REGISTERS AND FORCE "1" TO ESTR$_3$ OR ØSTR$_3$
ZDI$_{30-35}$ FOR TAG IS PRESENT BUT NOT VALID UNTIL $DA FROM MEMORY
WHAT IS TAG ON ZDI LINES?
ITS
ITB
IC OR $\overline{IC}$ CONDITION OF IC F/F
$\overline{IC}$
IC
$TBR ITS AS A RESULT OF $DA RECEIVED
$ESTR ITB AS A RESULT OF $DA RECEIVED
$ØSTR ITB AS A RESULT OF $DA RECEIVED

FIG. 10.2.2.1 BASE FRAME OPERATIONS FLOWCHART FOR APPENDING PI; APPENDING PA, PZ, PN, PT, AND WFM OPERATIONS

10.2.2.2 Appending Instructions (PI)

The first cycle that the Base Frame would normally go through is the prepare cycle for an instruction. This cycle is started when a ¢Set PI level is present from the Control Frame at the time that the Control Frame issues a $C to start the prepare cycle for the instruction address. Strobe $C sets a flip-flop in the Base Frame that indicates the PI cycle (¢Set PI present) which turns on the control point to place the PBR on the ZBRX bus to the Interface Frame and inhibits any ABR selection. By definition the PBR always contains the Pointer to the base of the Procedure Segment where the instruction is contained. The Pointer remains on the ZBRX bus until the next $C is received from the Control Frame to reset the PI flip-flop and close out the cycle.

10.2.2.3 Appending for Operands (PA, PZ, PN, PT)

Appending for operands when the Control Frame is in a PA, PZ, PN, or PT cycle is essentially the same as for a PI cycle, that is, sending an address pointer to the Interface Frame. However, in this case the pointer is to an external segment and the selection is more sophisticated. Selection may involve the addition of an internal base to the effective address in the Control Frame, in addition to the external base designated by the instruction, or by the ABR designated by the instruction.

10.2.2.3 - contd

Consider the "normal" case and assume that all required registers have been initialized by the Operating System and contain address pointers to specified segments and pages of segments of a users program. The first $C from the Control Frame sets a PA cycle flip-flop and also sets an IC flip-flop if the Control Frame is working on an odd instruction; an even instruction causes the IC flip-flop to remain reset. Since we are assuming that this is the first prepare cycle for the address, no ITS tag can be present. When bit 29 of the instruction word is a "1," then the address specified is assumed to consist of a segment tag (bits 0-2 of the address) and a signed 15 bit number to be used in finding the final address. If bit 29 of the instruction word is not set to a "1," then the address specified in the instruction word is assumed to be 18 bits, is located in the segment that is currently in execution and the PBR is gated onto the ZBRX bus.

When bit 29 of the instruction word is set to "1," the even or odd segment tag register (ESTR or OSTR) is decoded to select the ABR specified by the instruction word. If this ABR is defined as external (ABR_{21} = 1) meaning external to the segment currently specified by the PBR, then the external ABR designated by the instruction address field is switched to the ZBRX bus. If the ABR selected is internal (ABR_{21} = 0) then two things happen: the selected ABR (designated ABRn for internal) is switched to the ZBRI (internal) bus to the Control Frame, and bits 18-20 of the selected ABRn are placed on the ZXBN bus to another set of selection gates for the eight ARBs. Bits 18-20 of the internal base specify the linked external base, and when decoded put the linked external base on the ZBRX bus to the IF. The reason for this method of selection is because by definition an internal ABRn may not be selected without specifying a linked external base, but an external ABRm may be specified directly. This philosophy is a part of the total system hardware software requirements.

10.2.2.3 - contd

If the Control Frame was in the PA first prepare cycle for an operand, and bit 29 of the instruction word was a "1," then an internal base must be added to the address in the BS-adder in the Control Frame in addition to the external base that is sent on the ZBRX bus to the Interface Frame. What all this selection of ABRs or bases means; is that the address pointer to the instruction operand, is either selected by the tag portion of the address in the instruction word (external base to ZBRX bus), or bits 18-20 of the internal ABR (ZBRI bus to pick the external ABR); the internal ABR having been designated in the address field of the instruction word.

The output on the ZBRX bus is also present at the input to the TBR. This means that the TBR always contains the Pointer that is currently on the ZBRX bus. If a satisfied conditional transfer is indicated by the Control Frame ($TRGO_1$.EA present) then the pointer in the TBR is transferred into the PBR. The PBR now contains the pointer to the new Procedure Segment. When the next \$C is received from the Control Frame, the prepare cycle flip-flops may be changed and \$TBR and \$PBR may be generated.

10.2.2.4 Wait for Modifier (WFM)

The WFM cycle is entered when the Control Frame is waiting for an even indirect word that is the result of an RI/IR modification on the original or previous indirect word currently being processed. The $C that closes out the prepare cycle for the word being processed sets an RI/IR flip-flop indicating that RI/IR was the modification tag that is resulting in an indirect word fetch from memory. When strobe $PIN is received from memory indicating that an interrupt has been honored, the WFM flip-flop is set and the Base Frame is prepared to strobe the even or odd segment tag registers for an ITB tag, or the TBR for an ITS tag. If an ITB tag is present on the ZDI lines at the time that a $DA is received from memory, then the state of the IC flip-flop determines whether the even or odd segment tag register is strobed and the ITB tag flip-flop is set. When an ITS tag is decoded then the TBR is strobed with ZDI_{0-17} (the new address pointer) and the ITS tag flip-flop is set. Strobe $DA also results in resetting the WFM flip-flop effectively closing out the WFM cycle for the Base Frame.

When the prepare cycle for the indirect word just brought in is started by the next $C from the Control Frame, then the RI/IR flip-flop is reset. The ITB or ITS tag flip-flops remain set during the next prepare cycle started by $C. Resetting of the ITB flip-flop is dependent on the second $C that closes the next prepare cycle after an ITB tag is received; provided a Restore Control Unit instruction and effective address is not present at $C time. Resetting of the ITS tag flip-flop is accomplished when a final effective address for an operand is obtained, and a Restore Control Unit instruction is not present. What this means is that for ITB modification the segment tag register specifies the ABR containing the new pointer and is not changed until the instruction is executed and two new instructions are brought into the instruction register in the Control Unit.

10.2.2.4 - contd

Therefore the ITB tag flip-flop is reset after the first prepare cycle following the WFM cycle since it is only used to switch the internal base to the ZBRI bus for the first prepare cycle. The ITS tag flip-flop on the other hand is only reset when a final effective address is present, since the pointer in the TBR must be present on ZBRX to obtain the final address.

10.2.2.5 General Base Operations

When an instruction is being processed by the Control Unit that involves changing the Base Frame registers, or effecting operations related to the Base Frame, then strobe $BORY is received from the Interface Frame initiating a general base operation (GBO). Strobe $BORY generates strobe $B, the internal strobe for a general base operation, which in turn sets a GBO flip-flop. From this point on the specific operations performed by the Base Frame are dependent on the instruction being executed. Base Frame instructions are listed on G.E. drawing 43D139962. Generally speaking, the following actions occur:

1) $BORY is received, generating $B and setting the GBO flip-flop.

2) The proper control points are turned on for the specific instruction (see Base Frame "Spec Chart" 43D139962).

3) Control of the strobes for the various registers is determined by the specific instruction, Master/Slave processor mode and lock base bit conventions on the ABRs.

10.2.2.5 - contd

4) If the instruction involves loading any of the ABRs (specifically ADB_n, EAB_n, EAP_n, LBR_n, LDCF, or TSB_n) then a fault strobe ($FLT) is generated if the Master/Slave lock base conventions are not met (trying to load an ABR in Slave mode when the lock base bit is set).

5) The generation of a strobe $BC indicating that base operations are complete is sent to the Control Frame. Again, the generation of $BC is dependent on the specific instruction. Strobe $BC is used to generate a $C closing out the current cycles in Control and Base Frames.

In general, refer to the "Spec Chart" for the specific operation involved, and the description of the instructions in Part I of this EPS. The detailed discussion in paragraph 10.3 for some representative instructions indicate the specific actions that occur.

10.2.3 OPERATION OVERLAP

In discussing each basic operating cycle the reader may have gotten the impression that the cycles were distinctly separated from one another; such is not the case. As an example, consider the various cycles that take place for an instruction and operand fetch that involves a general base instruction.

A strobe $C is received from the Control Frame to initiate appending for an instruction (PI), the contents of the PBR are put on the ZBRZ bus to the IF Frame, a second strobe $C is generated, and the instruction word is fetched. When the Control Frame is going for an operand fetch the next $C that initiates a PA cycle ultimately results in the contents of an external ABR_m or the TBR being put on the ZBRX bus to the IF Frame. However, if the instruction fetched

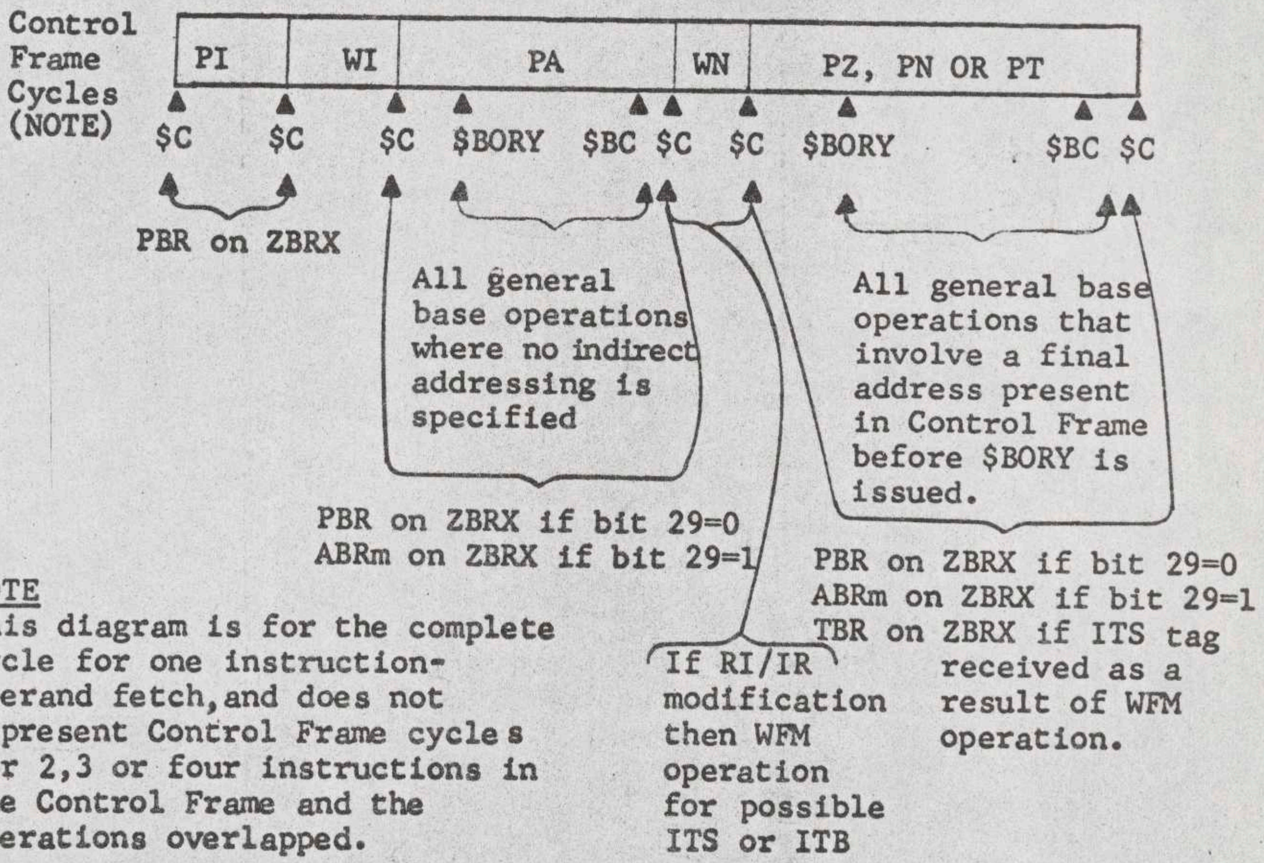
---

10.2.3 - contd

during the previous PI cycle involved a Base Frame operation and no RI or IR modification was involved, then the Control Frame stops and the Interface Frame issues a strobe $BORY to the Base Frame to start a general base operation. When a general base operation is started, the append flip-flop outputs are inhibited (the actual inhibit varies with the EAB_n and EAP_n instructions), but not reset. Only the second $C closing the PA cycle can reset the append flip-flop. With the generation of the $BC that indicates that a base operation is complete, the inhibits are removed. The Base Frame then returns to a "normal" appending for operands cycle until the next $C to close the current PA cycle resets the append flip-flop. This strobe $C is immediately after a $BC is sent to the Control Frame.

Operations within the Base Frame for general base operations may involve multiple access to memory before completion of the specific operation. In these cases the core location to go to, or fetch data from is formed prior to the Interface Frame issuing a strobe $BORY. The pointer on the ZBRX bus is strobed into the address register in the Interface Frame (via the RS-adder) concurrent with the receipt of a $BORY. This pointer was selected as soon as a $C is received from the Control Frame as previously explained. When any modification is specified in the original or indirect word, then strobe $BORY is not received until the final address fetch. What this means is that a general base operation cannot be started unless the Control Frame is in the cycle for a final address fetch. Fig. 10.2.3 illustrates the various cycles and how they overlap showing only relationships not actual timing, and using $C as a reference.

10.2.3 - contd


Control Frame Cycles (NOTE)
PI
WI
PA
WN
PZ, PN OR PT
$C
$C
$C
$BORY
$BC
$C
$C
$BORY
$BC
$C
PBR on ZBRX
All general base operations where no indirect addressing is specified
PBR on ZBRX if bit 29=0
ABRm on ZBRX if bit 29=1
All general base operations that involve a final address present in Control Frame before $BORY is issued.
PBR on ZBRX if bit 29=0
ABRm on ZBRX if bit 29=1
TBR on ZBRX if ITS tag received as a result of WFM operation.
If RI/IR modification then WFM operation for possible ITS or ITB

NOTE
This diagram is for the complete cycle for one instruction-operand fetch, and does not represent Control Frame cycles for 2,3 or four instructions in the Control Frame and the operations overlapped.

Fig. 10.2.3 Base Frame Operating Cycles in Relation to Control Frame Cycles.

10.2.4 STROBE GENERATION

Strobe generation within the Base Frame may be generalized into two categories: external and internal. The external strobes received are from the Control Frame to initiate normal appending operations for instructions and operands ($C, $PIN); and from the Interface Frame to initiate and control general base operations ($BORY, $DA, $DS). Internal strobes ($B) to sequence operations within the Base Frame are only used during general base operations. The internal strobes are generated as a function of one or all of the external strobes from the Interface Frame ($BORY, $DA, $DS) and the specific Base Frame instruction being executed. The one exception is strobe $DA which is used to set the ITS or ITB flip-flops when in WFM cycle.

10.2.4.1 External Strobe Utilization

The five external strobes are utilized as follows.

1) Strobe $C -- reamplified and used to initiate or terminate the normal appending cycle for instructions and operands, and if an RI/IR modifier is specified by the tag of the instruction word, set the RI/IR flip-flop in preparation for a WFM cycle.

2) Strobe $PIN -- reamplified and used to set the WFM flip-flop in anticipation of the receipt of an ITS or ITB modifier in the even indirect word of a pair pulled from memory as a result of RI/IR modification to an even memory address.

3) Strobe $BORY -- used to initiate the internal $B chain for general base operations. $BORY is never received from the Interface Frame except when a final effective address is present in the Interface Frame.

10.2.4.1 - contd

4) Strobe $DA -- delayed to obtain $\$DA_{\Delta}$ to compensate for pulse versus d-c propagation delay in the data lines from memory. Strobe $\$DA_{\Delta}$ with a small compensating delay is used to strobe the ITB and ITS flip-flops for WFM cycles and initiate an internal $B for general base operations. When executing the LDCF and ADB_n instructions, $\$DA_{\Delta}$ is delayed approximately 250 nanoseconds further to allow the BS-adder in the Control Frame to stabilize before initiating a $B.

5) Strobe $DS -- received from the Interface Frame as a result of double precision store operations effecting the base frame registers and the increment counter. Strobe $DS's signify that data is stored in the Data-Out Register in the Interface Frame and initiate a $B as does $BORY and $DA.

10.2.4.2 Internal Strobe Utilization

The internal strobe chain consists of a gated-amplifier delay line loop that is initiated by $\$BORY + \$DA_{\Delta} \cdot GBO + \$DS \cdot GBO$ to generate $\$B_{10}$, $\$B_{11}$, $\$B_{13}$, and $\$B_{\Delta}$ for use throughout the Base Frame during general base operations. Any-one of the three strobes mentioned can generate one set of strobe $B's ($\$B_{10}$, $\$B_{11}$, $\$B_{13}$, and $\$B_{\Delta}$). For instructions TSB_n, LDCF, CAM, EAB_n, EAP_n, or a load base fault, a feedback loop is enabled and additional $B's are generated to finish the various operations required. The $B's are used to step the increment counter, generate (subject to restriction) $\$ABR_{0-17,18-21,23,22}$; $PBR; $TBR; $ITB; $ITS; $BRY; $FLT and $DP throughout the base frame. The only difference between $B's is for the compensating delay required for circuit timing.

10.2.5 INCREMENT COUNTER

Many of the instructions classified as general base instructions involve more than one specific operation. To keep track of the number of the specific operation , an increment counter is used. This counter is a serial binary counter that is incremented every time $\$B_{13} \cdot \text{GBO} \cdot \overline{\text{¢}\$\text{BC}}$ is true. The parallel outputs of the increment counter are decoded for counts of 0, 1, 2, 5, 7, and 31. Counts of 0, 1, 5, 7, and 31 are used for instructions that take 0, 2, 6, 8, and 32 steps respectively. The increment counter is not incremented with the first or last $\$B_{13}$ received, and only contains the number of operations that have been completed. The different instructions by number of operations are:

1) One step - LBRn, EABn, ADBn, SBRn, LDBR, SDBR, CAM

2) Two step - EAPn, LDCF, RTD, STCD, STPn, SAZ, TSBn

3) Six step - SCU, RCU

4) Eight step - LDB, STB

5) Thirty-two step - SAM

When the final $BC is generated to complete the general base operations cycle, the GBO flip-flop is reset resetting the increment counter.

10.2.6 LOAD BASE FAULT DETECTION

Fault detection circuits are included for those instructions that involve changing an address base register: ADBn, EAB, EAP, LBRn, LDCF, and TSBn. No fault is generated for the LDB instruction. Generation of the fault is a function of the selected ABR, a general base operation, the specific instruction and the Master/Slave lock base bit restrictions (all load base instructions may only be executed in Master mode of if $ABR_{22} = 0$). If an attempt is made to change the specified ABR when not in Master mode, and the lock base bit (ABR_{22})="1", then a ¢Set FLT level is generated which inhibits the generation of a $BC and permits the generation of a $FLT on receipt of the next $\$B_{13}$.

10.2.7 EVEN AND ODD SEGMENT TAG REGISTERS (ESTR And ØSTR)

The even and odd segment tag registers as the names imply are used to hold the segment tag and bit 29 of the instruction word for the even (ESTR) and odd (ØSTR) instructions in the Y- and C-registers of the Control Frame. In addition, if an ITB modifier is received as a result of a WFM Base Frame operation, the new segment tag (designating ABR 0 through 7) is strobed into ESTR or ØSTR. The strobes are received concurrent with the strobes for the Y- and C-registers in the Control Frame for normal appending on instructions, and for an RCU instruction. Strobes for the ITB modifier received tag are generated internal to the Base Frame as a result of the $DA received during the WFM cycle. With ITB modification a "1" is forced into $ESTR_3$ or ØSTR_3 to simulate bit 29 = 1 of the instruction word.

10.2.8 DATA IN (ZDI) SWITCH

All data from all eight possible Memory Controllers to the Processor is received at Ports A-H of the Base Frame for distribution to other areas within the Processor. Port selection signals from the Interface Frame determine which port is enabled at which specific time. Associated with the ZDI switch is a 18 bit buffer for direct operands from the Control Frame to the OU. This is because data distribution of the direct operand for DU, or DL modification is also on the ZDI bus to the Register Frame of the OU.

10.3 DETAILED OPERATION (see GE drawings 43D1060212, sheet 9, and 43D163422)

10.3.1 APPENDING INSTRUCTIONS (PI)

Appending for Instructions (PI) takes place whenever the Control Frame generates DSPI1B (¢ Set PI) and QSC01B ($¢) on the input lines to the Base Frame. Appending takes place whether or not the Processor is in the Absolute or Append mode of operation. The decision on whether or not to use the information present on the DX__F(ZBRX) bus is a function of the Interface Frame and the Control Frame. Whether or not to use the information present on the DN__F(ZBRI) bus is a function of the Base Frame. Refer to the timing diagram (Fig. 10.3.1.).

The sequence of events that take place begins with the presence of DSPI1B (¢ Set PI) from the Control Frame prior to the receipt of QSC01B ($C). When $C is received, flip-flop EPIOF (PI) is set. The set output of the PI flip-flop is the BRXPOF (¢PBR$_{0-17}$/ZBRX$_{0-17}$) control level that places the contents of the Procedure Base Register (PBR) on the ZBRX bus to the Interface Frame via the ZDO switch, and blocks the control level generation for the eight address Base Registers; specifically, BRXOF ($\overline{\text{¢\$EX}}$). Since by definition the Procedure Base Register contains the address of the base of the Procedure Segment, and all instructions will be contained in the Procedure Segment, only PBR$_{0-17}$ is required of the Base Frame during PI cycles and address appending. When the second $C is received from the Control Frame the PI flip-flop is reset and the control level (BRXPOF) to turn on the PBR/ZBRX switch is removed.

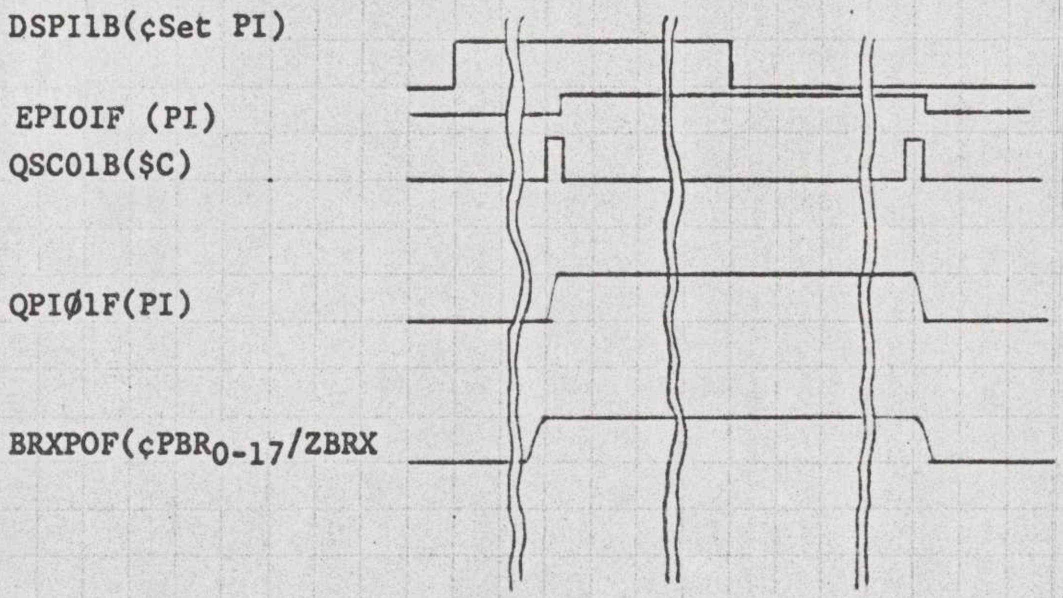

DSPI1B(¢Set PI)
EPIOIF (PI)
QSC01B($C)
QPI01F(PI)
BRXPOF(¢PBR0-17/ZBRX

Fig. 10.3.1 Appending Instructions (PI),
Timing Diagram

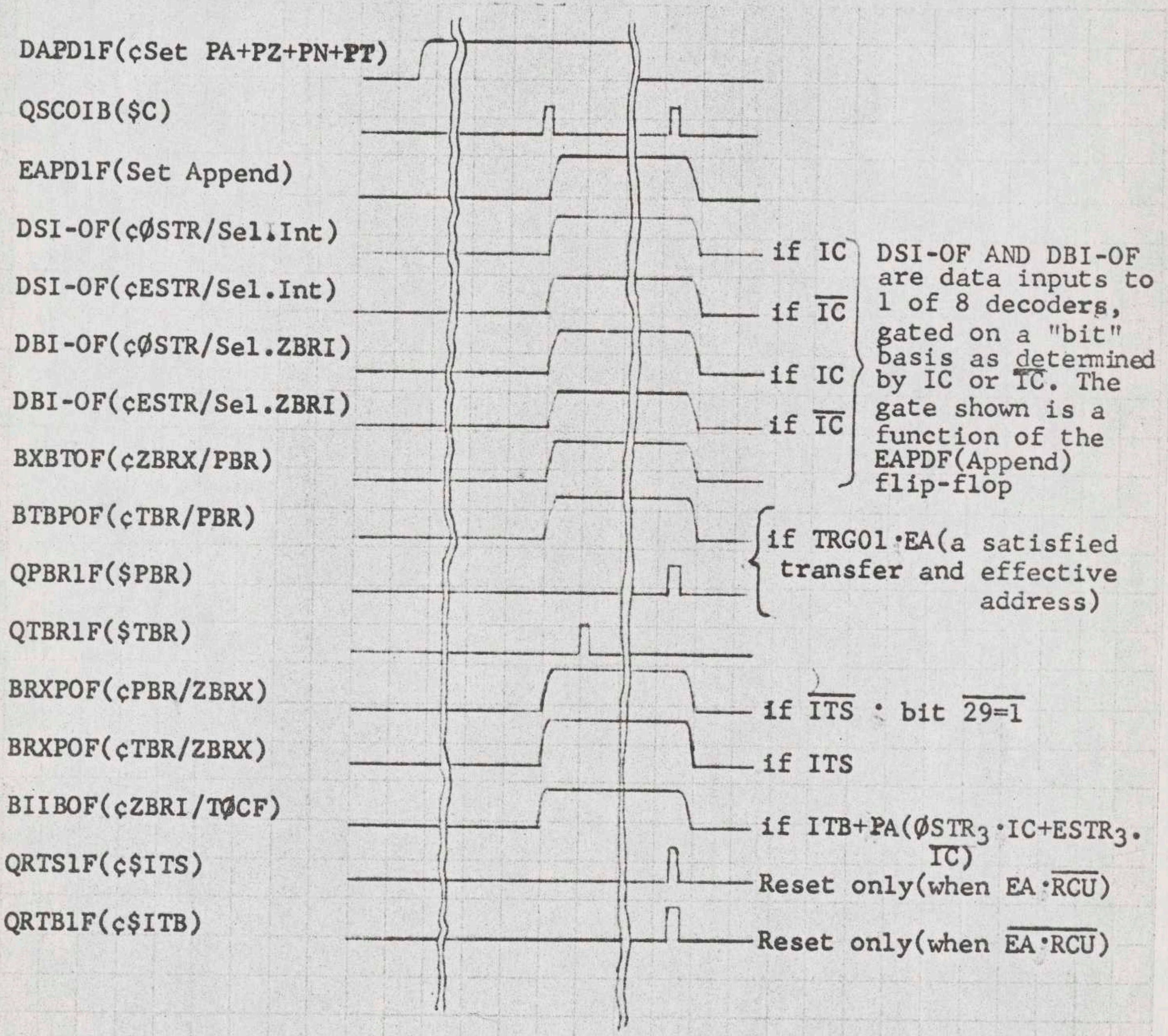
10.3.2 APPENDING OPERANDS (PA, PZ, PN, PT)

Address appending for operands is essentially the same for PA, PZ, PN, or PT cycles. As in appending for instructions, the Base Frame goes through the appending process whether the Processor is in Absolute or Append mode of operation. Refer to the timing diagram (Fig. 10.3.2) during the following discussion.

Control level DAPD1F (¢ Set PA+PZ+PN+PT), resulting from levels sent from the Control Frame in conjunction with QSC01B($C), set the EAPDF flip-flop indicating that appending is required. With the EAPDF flip-flop set, the even or odd segment tag register is switched to the one of eight decoders for both internal and external base register selection. Which segment tag register is switched is determined by the EICOF(IC_1 and IC_2) flip-flops. When register (ØSTR) is selected; reset causes the even segment tag register (ESTR) to be selected.

10.3.2.1 Internal Base Selection

Selection of an internal base is accomplished when the selected segment tag register at DBI__F is decoded to pick one of the eight address base registers that has been designated an internal base by having its internal/external base bit (bit 21) set to 0. Control level BRN__OF which selects the register is not generated unless F+21OF (complement of bit 21 of the ABR designated by the segment tag register) indicates that the selected register is internal. The selected base register is put on the ZBRI bus to the Control Frame and ultimately the BS-adder, via the $\overline{\text{ZEROS}}$ switch. This switch is only open during the PA cycle of an appending operation with bit 29 of the instruction word set, for an ITB modifier during an appending operation, and for EABn and EAPn instructions during general base operations. At other times, 0's are sent to the BS-adder in the Control Frame instead of the selected internal base.


DAPD1F(¢Set PA+PZ+PN+PT)
QSCOIB($C)
EAPD1F(Set Append)
DSI-OF(¢0STR/Sel.Int)
DSI-OF(¢ESTR/Sel.Int)
DBI-OF(¢0STR/Sel.ZBRI)
DBI-OF(¢ESTR/Sel.ZBRI)
BXBTOF(¢ZBRX/PBR)
BTBPOF(¢TBR/PBR)
QPBR1F($PBR)
QTBR1F($TBR)
BRXPOF(¢PBR/ZBRX)
BRXPOF(¢TBR/ZBRX)
BIIBOF(¢ZBRI/T0CF)
QRTS1F(¢$ITS)
QRTB1F(¢$ITB)
if IC
if $\overline{IC}$
if IC
if $\overline{IC}$
if $\overline{ITS}$: bit $\overline{29=1}$
if ITS
if ITB+PA($0STR_3 \cdot IC+ESTR_3 \cdot \overline{IC}$)
Reset only(when $EA \cdot \overline{RCU}$)
Reset only(when $\overline{EA \cdot RCU}$)

DSI-OF AND DBI-OF are data inputs to 1 of 8 decoders, gated on a "bit" basis as determined by IC or $\overline{IC}$. The gate shown is a function of the EAPDF(Append) flip-flop

if TRG01·EA(a satisfied transfer and effective address)

Fig. 10.3.2 Appending Operands (PA, PZ, PN, PT), Timing Diagram

---

10.3.2.2 External Base Selection

The output of the selected tag register (ESTR or OSTR) at DSI_1F is decoded by the one of eight octal to decimal decoder to select one of the eight ABR's (control points DRX_1F). If the selected ABR contains an external base, as signified by a 1 in bit 21 (F_211F) then level BEXTOF is generated and the (BXXX1F) input becomes one of the enable inputs to one of the two ABR selection gates associated with the ABR that generates BRXOF (¢$EX)--the control of ABR$_{0-7}$ to ZBRX. In this particular case, $\overline{(\text{¢TBR}_{0-17}/\text{ZBRX}_{0-17} + \text{¢PBR}_{0-17}/\text{ZRX}_{0-17})}$(BXXX1F)(DRX-1F), for the register selected, generates $(\overline{\text{¢SEX}})$. The complement of bit 21 (0), of the register selected is used to generate level BNNNOF which inhibits the selection of another ABR via the NAND gate associated with the DXB_1F (ZXBN) bus. Bus ZXBN is the internal base bus that is decoded for the selection of the associated external base if an internal base is specified by the segment tag register. The philosophy here is that an internal base designated by the segment tag register must specify in bits 18-20 of its control field, the associated, or linked external base. If an external base is specified by the segment tag register no associated internal base can be specified and none will be put on the ZXBN bus.

If bit 21 of the ABR selected by DRX_1F is 0, indicating that an internal base is specified by the segment tag register, bits 18-20 of the selected register are switched on the ZXBN bus, BNNNOF is enabled and BXXX1F disabled. The NAND gate associated with the selection of an external base when specified by the segment tag register is now disabled. When bit 21 of the selected ABR is 1 indicating an external base then $\overline{\text{¢\$EX\_}}$ is a function of $\overline{(\text{¢TBR}_{0-17}/\text{ZBRX}_{0-17} + \text{PBR}_{0-17}/\text{ZBRX}_{0-17})}$(BNNNOF)(ZXBN$_{0-2}$), the

10.3.2.2.- contd

NAND gate output that is associated with internal/external base selection. When performing the base instructions STB and SBRn with the GBO flip-flop set, BXXX1F is permanently enabled and BNNN0F disabled to force all base registers, when selected by the one of eight decoder, on the ZBRX bus regardless of whether the ABR's are designated internal or external.

The output of the selected external base register (or the PBR during periods when bit 29 of the instruction word = 0) on the ZBRX bus is gated to the input of the Temporary Base Register by control level BXBT0F. The data is settled by the time QTBR1F ($TBR) strobes ZBRX$_{0-17}$ into the TBR (approximately 290 nanoseconds after $C). In addition to ZBRX$_{0-17}$ being strobed into the TBR it is placed on the DD__0F (ZD0) switch bus to the Interface Frame. Control of BXB00F for the ZBRX portion of the ZD0 switch during PA, PZ, PN, and PT appending operations is dependent on BSN01B ("Snapshot 0") from the Control Frame, that is, when a "snapshot" is taken as the result of a recognized fault ZBRX is not sent to the Interface Frame.

Transfer of Control

When a satisfied transfer of control has been determined by the Control Frame, both BEAC1B (EA) and BZDC1B (TRG01) are present in the Base Frame. This causes BTBP0F (¢TBR$_{0-17}$/PBR$_{0-17}$), the control point for placing TBR into PBR, to be enabled. Concurrent with the reception of the next $C from the Control Frame, QPBR1F ($PBR) is generated strobing the contents of TBR into PBR; $PBR occurs after the new external base had been strobed into the TBR. After PBR has been loaded with the new external base the normal instruction address preparation sequence for PI is started by the Control Frame.

ITS Modifier

If an ITS modifier is detected as the result of a previous WFM Base Frame operation; EITS1F(ITS)· EAPD1F(Append)· EGBOOF($\overline{\text{GBO}}$) cause BRXTOF ($¢TBR_{0-17}/ZBRX_{0-17}$) to switch the TBR onto the ZBRX bus and inhibit the selection of an external base from the eight ABR's. Level $¢TBR_{0-17}/ZBRX_{0-17}$ generation is dependent on the append level generated by (PA+PZ+PN+PT)·$\$C$ received from the Control Frame; and consequently places TBR_{0-17} on the ZBRX bus only in the time internal between $\$C$'s. The TBR had been loaded previously with the new segment base pointer from ZDI_{0-17} during the WFM Base Frame cycle.

ITB Modifier

When an ITB modifier is detected as the result of a previous WFM Base Frame operation, the even or odd segment tag register contains the tag to the new external ABR. As a result, the external base is selected in the normal manner as previously described for PA, PZ, PN, and PT operand fetches.

10.3.2.3 ITS and ITB Reset

Reset of the EITBF(ITB) flip-flop is accomplished when ($\overline{\text{RCU}\cdot\text{EA}}$)(PA+PZ+PN+PT)($\$C_{11}$) is satisfied at the end of the prepare cycle for an operand where an ITB modifier is specified. The first $\$C_{11}$ received that started Base Frame operations does not reset the ITB flip-flop since the PA, PZ, PN, PT(EAPDF) flip-flop input is not present till after the first $\$C_{11}$. This is due to built in delay from $\$C11$ time to the receipt of the PA, PZ, PN, PT level received at the ITB flip-flop reset input.

10.3.2.3 - contd

The EITSF (ITS) flip-flop is reset in the same manner as the ITB flip-flop but when $(\overline{RCU}\cdot EA)$ $(\$C_{11})$ is satisfied. This condition exists only when the final effective address (EA) for the operand is prepared by the Control Frame and no RCU instruction has been decoded by the Control Frame.

Differences in the resets are because in ITB modification the new external base tag has been strobed into the applicable segment tag register, and base selection is performed normally for the first prepare (PA) cycle. The ITB flip-flop level must be "up" in anticipation of an internal base being specified by the new segment tag register (and consequently, ABR) contents. The level indicating an ITS modifier, on the other hand, must be "up" to allow the ¢TBR/ZBRX control level to keep the contents of the TBR on the ZBRX bus until a final effective address is prepared; TBR containing the new address pointer as previously specified.

10.3.3 WAIT FOR MODIFIERS (WFM) OPERATION

The third specific operation that the Base Frame performs is to prepare for the receipt of an ITS or ITB modifier in an even indirect word of a even odd pair pulled as a result of RI/IR modification, and to load the received data present on the ZDI lines from memory into the applicable segment tag register (ITB modification), or temporary Base Register (ITS modification). See timing diagram (Fig. 10.3.3) during the following discussion.

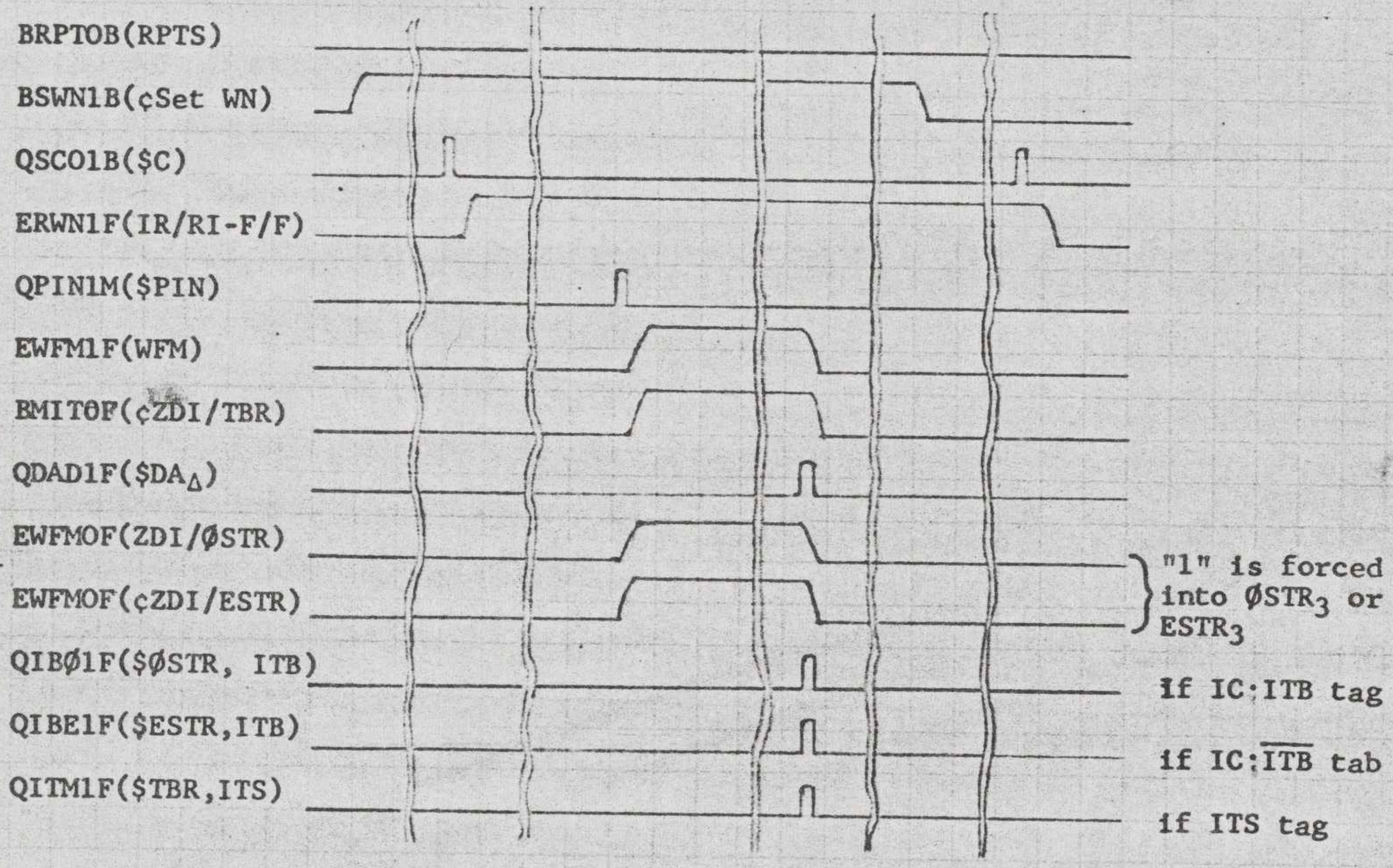

BRPTOB(RPTS)
BSWN1B(¢Set WN)
QSC01B($C)
ERWN1F(IR/RI-F/F)
QPIN1M($PIN)
EWFM1F(WFM)
BMIT0F(¢ZDI/TBR)
QDAD1F(DA_\Delta$)
EWFM0F(ZDI/0STR)
EWFM0F(¢ZDI/ESTR)
QIB01F($0STR, ITB)
QIBE1F($ESTR,ITB)
QITM1F($TBR,ITS)
"1" is forced into 0STR$_3$ or ESTR$_3$
if IC:ITB tag
if IC:$\overline{\text{ITB}}$ tab
if ITS tag

Fig. 10.3.3 Wait For Modifiers (WFM) Operations, Timing Diagram

---

10.3.3 - contd

When the Control Frame sends (BSWN1B)(DW131B)(MB170B)(BRPTOB) representing ¢ Set WN, ZI13, $\overline{\text{BS17}}$, and RPTS respectively to the Base Frame, it indicates that it is going to ask for an even indirect word that is the result of an RI/IR modification tag in the instruction or indirect word. With the foregoing levels from the Control Frame the next $C sets the ERWNF (RI/IR) flip-flop. ERWN1F supplies the enable to the EWFMF (WFM) flip-flop and WFM is set upon receipt of the next QPIN1M($PIN) from memory indicating that a Processor interrupt is acknowledged. Concurrent with the setting of WFM, control levels BMITOF (¢ZDI_{0-2} OSTR_{0-2},1/OSTR_3; ¢ZDI_{0-2}/ESTR_{0-2}, 1/ESTR_3) are generated. When QSDA1M($DA) is received from the Interface Frame indicating that data is on the ZDI lines for approximately 250 nanoseconds, QDAD1F($\$\text{DA}_{\Delta}$) is generated. $\$\text{DA}_{\Delta}$ is approximately 90 nanoseconds after $DA, and is used to set either the EITBF (ITB) or EITSF (ITS) flip-flops and reset the WFM flip-flop.

The ITB flip-flop is set when ZDI_{30-35} indicates an ITB tag in the even indirect word, WFM is set, IC or $\overline{\text{IC}}$ is up, and $\$\text{DA}_{\Delta}$ is received. WFM is present long enough after $DA so that the ITB flip-flop may be set. IC and $\overline{\text{IC}}$ determine whether QIB01F($OSTR ITB) or QIBE1F ($ESTR ITB) generates QITB1F setting the ITB flip-flop and resetting the ITS flip-flop; IC for $OSTR ITB, $\overline{\text{IC}}$ for $ESTR ITB. The ITS flip-flop is reset whenever ITB is set to eliminate the possibility of both ITS and ITB being set at the same time. This could occur if an ITS modifier was followed by an ITB before the ITS had a chance to reset. If an ITS modifier is detected on ZI_{30-35} at $\$\text{DA}_{\Delta}$ time the ITS flip-flop will be set and QITM1F($TBR ITS) generated.

10.3.3 - contd

For ITB modification OSTR is strobed if (IC)(ITB tag), ESTR is strobed for $(\overline{IC})$ (ITB tag), and the applicable segment tag register has the new segment tag. A "1" is forced into $OSTR_3$ or $ESTR_3$ when the new ITB modifier is received to indicate in later appending operations that base has to be selected using the segment tag registers.

For ITS modification, $TBR, ITS strobes the data on ZDI_{0-17} into the TBR. The TBR is used as the address pointer to the external base for ITS modification as mentioned previously.

10.3.4 GENERAL BASE OPERATIONS

The GE-645 Instruction Set contains 19 op codes (68 instructions) that directly affect operations that are peculiar to the Base Frame. Whenever these instructions are decoded by the Control Frame and a start pulse is received from the Interface Frame, the Base Frame enters the general base operation mode of operation. When the specific instruction execution is complete, a finish pulse is sent to the Control Frame to indicate that the Base Frame is complete.

10.3.4.1 Add to Base$_n$ (ADB$_n$) (See timing diagram Fig. 10.3.4.1)

The receipt of QBOR1M ($BORY) from the Interface Frame indicates that a general base operation (GBO) is to be started. $BORY generates QABR1F ($B) the general base strobe used throughout the Base Frame. Strobe $\$B \cdot \text{BCBCOF}(\overline{\text{¢SBC}})$ sets the GBO state flip-flop sending EGB01F to all circuits concerned with Base Frame operations. Level EGB01F is present during all base frame operations and is reset with the final $\$B_{13}$ that generates QZBC1F($BC), the finish pulse for base frame operations.

When the GBO flip-flop is set, the decoded ADB n instruction on the DW__B (ZI) lines from the Control Frame and GBO cause $\overline{\text{BADSOF}}(\text{¢}ZI_{2,7,8}/\text{Sel Int})$, BADIOF $(\text{¢}ZI_{2,7,8}/\text{Sel ZBRI})$, BIIBOF (¢ZBRI/toCF) and BBSBOF $(\text{¢}BS_{0-17}/ABR_{0-17})$ control levels to be generated. This switches the designated code for the effected ABR (as derived from the op code on ZI) to the one of eight decoders, opens the switch for the internal base bus to the Control Frame and switches the output of the BS-adder in the Control Frame to all eight ABR's (0-7). The decoded DSI_1F bits are used to generate BIN_0F(¢$INT_) which controls the selection of the strobe for the selected ABR. DBI_IF bits are used to select the internal ABR_{0-17} that is sent to the BS-adder in the Control Frame as previously explained. The ABR effected is not checked for internal or external base bit restrictions since the ADBn instruction allows an internal or external base to be specified.

When QSDA1M($DA) is received from the Interface Frame QDAD1F($\$DA_{\Delta}$) is generated and subsequently generates QABR1F($\$B_{13}$). Generation of QAB-1F($\$ABR_{0-17}$) is dependent on $(\$B_{13}) \cdot (GBO) \cdot (\text{¢}\$INT\_) \cdot$ BSL_1F (Master = 1 + Lock bit = 0). In other words the strobe for the applicable ABR is generated after the output of the BS-adder has had time to settle and the Processor is in Master mode or the lock base bit ($ABRn_{22}$) for the selected register is "0."

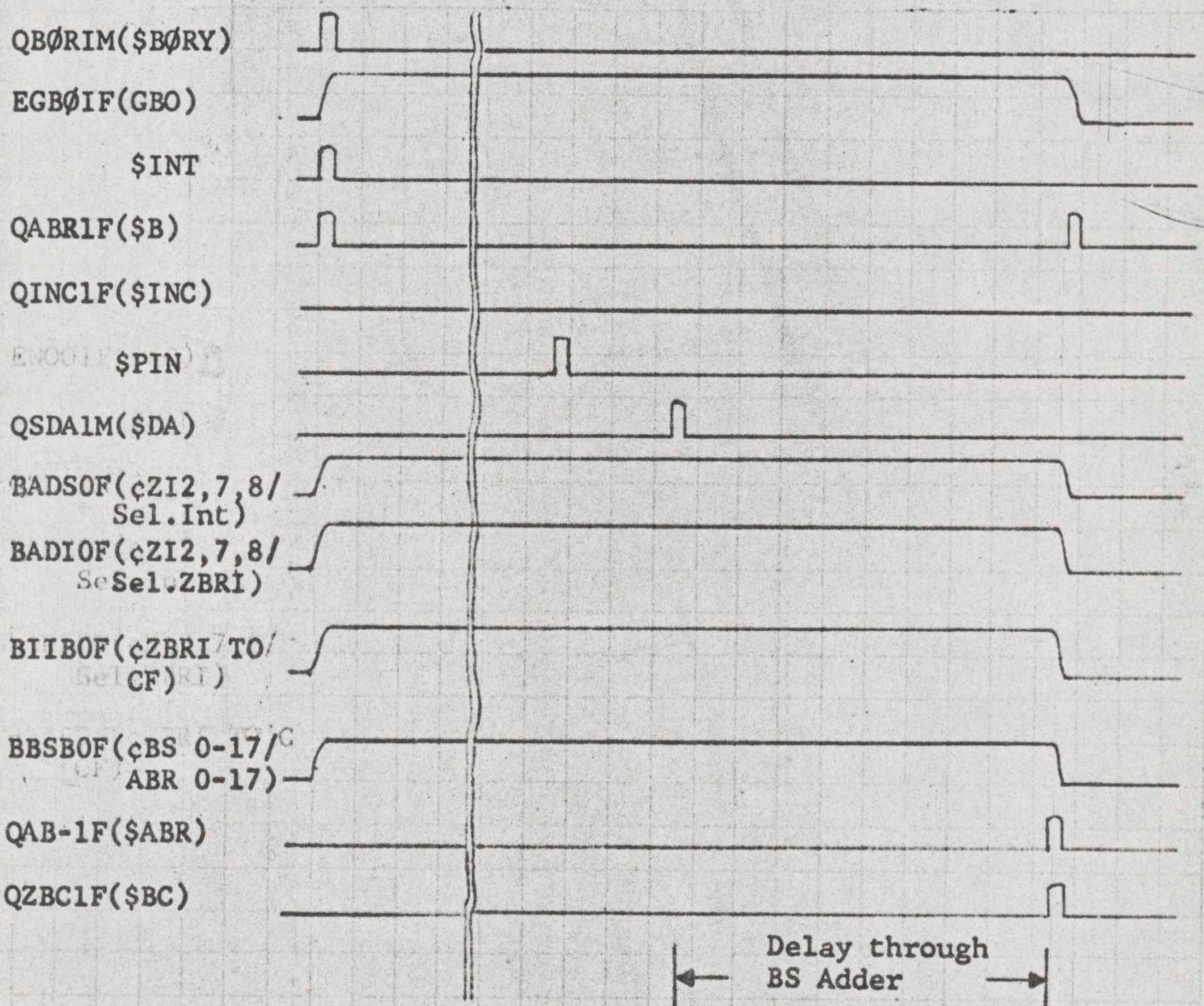

QB0RIM($B0RY)
EGB0IF(GBO)
$INT
QABR1F($B)
QINC1F($INC)
$PIN
QSDA1M($DA)
BADSOF(¢ZI2,7,8/ Sel.Int)
BADIOF(¢ZI2,7,8/ Sel.ZBRI)
BIIBOF(¢ZBRI TO CF)
BBSBOF(¢BS 0-17/ ABR 0-17)
QAB-1F($ABR)
QZBC1F($BC)
Delay through BS Adder

Fig. 10.3.4.1 Add to Base n (ADBn), Timing Diagram

10.3.4.1 - contd

Concurrent with the generation of $\$ABR_{0-17}$, $BC is generated indicating that Base Frame operations are complete. However, $BC is only generated if no "Fault" occurred in the selection of one of the ABR's (¢$INT_)--there was no attempt to change the selected ABR when the ABR was locked against change and the Processor was not in Master mode. Level BFLTOF (¢SetFLT) will always enable the $BC circuits unless a Fault has occurred.

10.3.4.2 Clear Associative Memory (CAM) (See timing diagram Fig. 10.3.4.2)

The Clear Associative Memory Instruction is used only to reset the empty/full bit and initalize the usage count bits in each Associative Register. Strobe $BORY from the Interface Frame generates $\$B_{13}$ which sets the GBO flip-flop and eventually sends BCAM1F (Reset Associative Memory) to the Associative Memory. A second $\$B_{13}$ is generated as a result of the recirculation loop in the $B chain being enabled by the BYCF1B(Y) term input to the second section of the $B chain that generates QIBD1F.

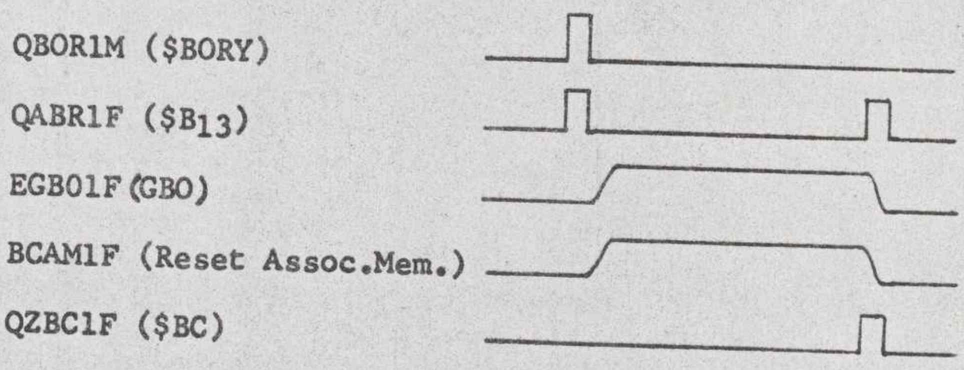


Fig. 10.3.4.2 Clear Associative Memory (CAM), Timing Diagram

10.3.4.3 Store Associative Memory (SAM) (See timing diagram Fig. 10.3.4.3)

The SAM instruction operation within the Base Frame is primarily concerned with keeping track of the number of registers stored by the IF frame and generating $BC to indicate that operations are complete.

Strobe $BORY received from the Interface Frame generates $\$B_{13}$, sets the GBO flip-flop generating BSAM1F (SAM·GBO to the Associative Memory) and then generates QSBD1F ($\$B_{\Delta}$). Strobe $\$B_{\Delta}$ is used in conjunction with EN000F-EN050F($\overline{\text{INCO}}$) · BBRY1F (enable for all stores) to generate QBRY1F($BRY) indicating that the Base Frame is ready. The term INCO is tied to the 2^0 position of the increment counter (EN01_F - EN05_F) indicating that the circuitry is prepared to count beginning with the even word location. When the Interface Frame issues $INT, QDST1M($DS) is received, generating another $\$B_{13}$ and QDPB1F($DP). Strobe $DP is a function of INCO (the 2^0 position of the increment counter which has been incremented one count by $\$B_{13}$) and indicates to the Interface Frame that this strobe is for the second half of a double-precision store for the odd word. The Interface Frame turns the $DP around and sends another $DS to the Base Frame thereby incrementing the increment counter to two and generating another $BRY. As before, $BRY indicates that the Base Frame is ready for the next store. However, the Interface Frame will have to wait until it has gained control of the memory and stored both words of data, as signified by a $DA received from the memory, before issuing another $INT and subsequent $DS to the Base Frame.

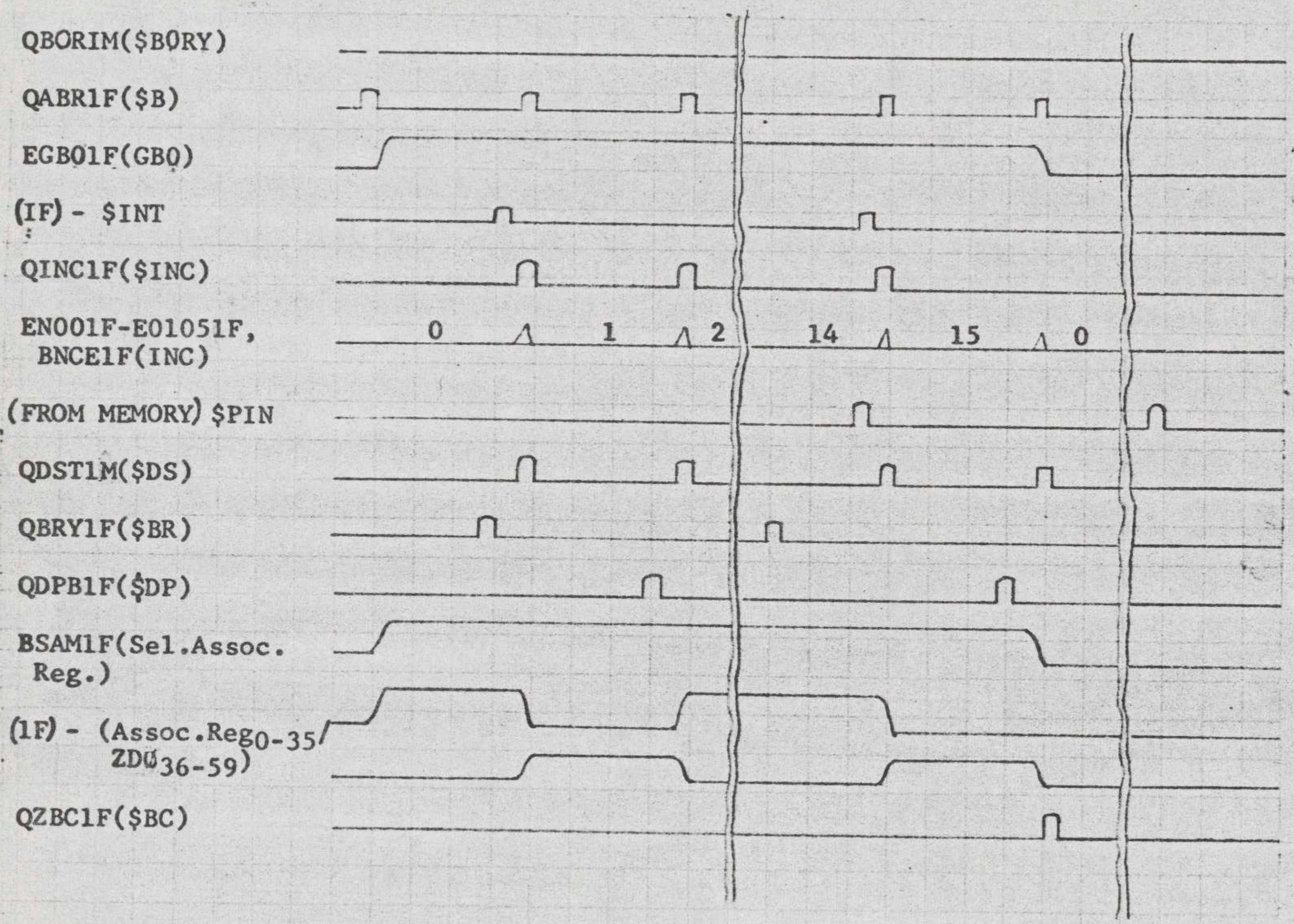

QBORIM($BORY)
QABR1F($B)
EGB01F(GBO)
(IF) - $INT
QINC1F($INC)
EN001F-E01051F,
BNCE1F(INC)
0
1
2
14
15
0
(FROM MEMORY) $PIN
QDST1M($DS)
QBRY1F($BR)
QDPB1F($DP)
BSAM1F(Sel.Assoc.
Reg.)
(1F) - (Assoc.Reg0-35/
ZD036-59)
QZBC1F($BC)

Fig. 10.3.4.3 Store Associative Memory Instruction (SAM), Timing Diagram

10.3.4.3 - contd

The above procedure is repeated for sixteen double-precision stores and 31 counts of the increment counter. With a count of 31 in the increment counter BCB1F(¢$BC)· BFLTOF(¢Set FLT) are enabled so that the next $\$B_{13}$ that is generated as result of the last $DS from the Interface Frame, causes $BC and resets the GBO flip-flop indicating that base operations are complete. The last $BRY is not generated since the removal of GBO from the $B circuits prevents generating the last $BΔ in addition to resetting the increment counter.

10.3.4.4 Store Associative Memory Zero (SAZ) (See timing diagram Fig. 10.3.4.4)

The operation performed by the Base Frame for the SAZ instruction is identical to the SAM instruction except for the following: BSAZ1F (SAZ·GBO) is sent to the Associative Memory instead of BSAM1F and only one double-word store is made before $BC is generated; strobe $BC now being a function of an increment count of 1·SAZ rather than 31·SAM.

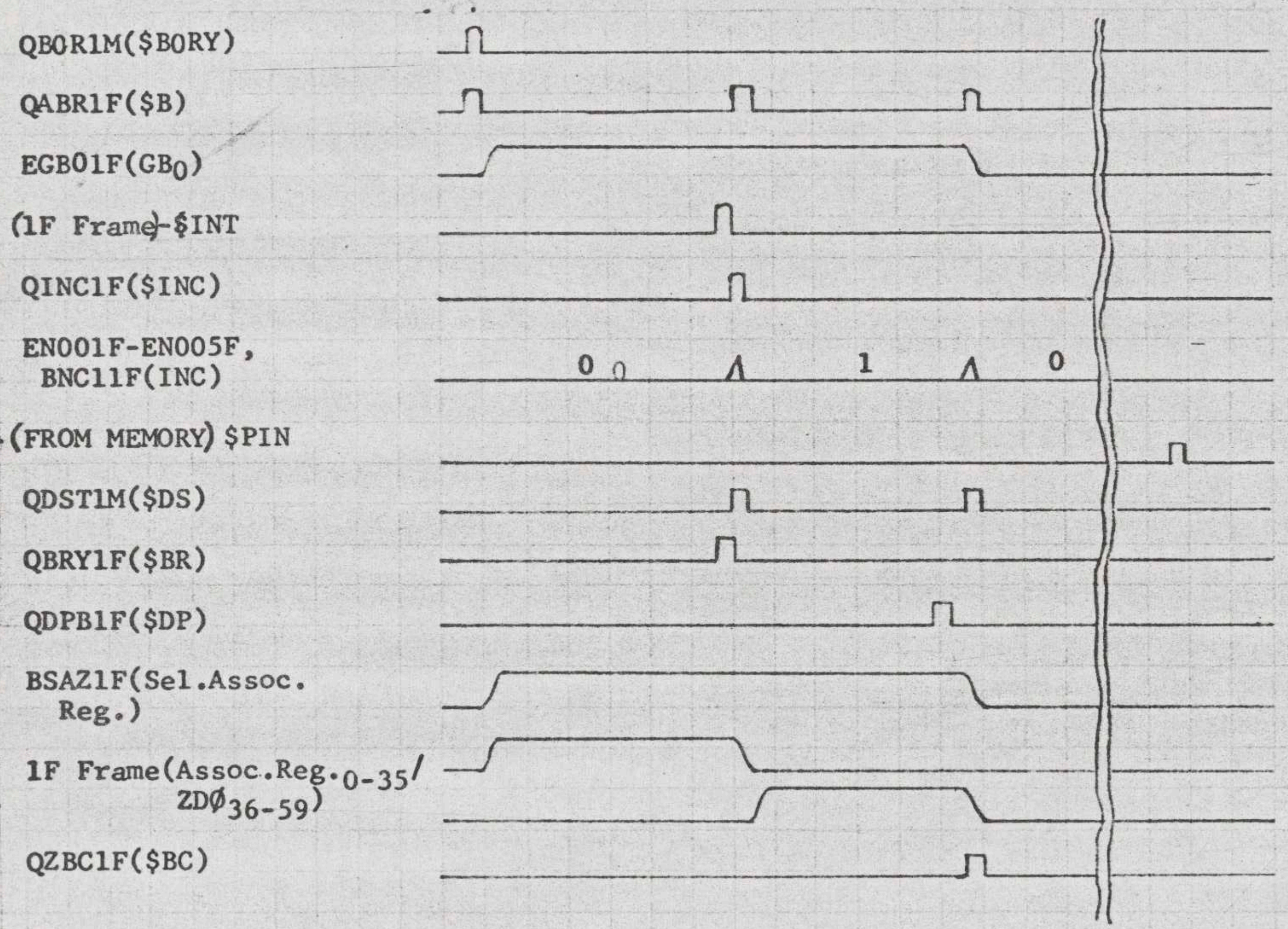

QBOR1M($BORY)
QABR1F($B)
EGB01F(GB_0)
(1F Frame)-$INT
QINC1F($INC)
EN001F-EN005F,
BNC11F(INC)
0
0
1
0
(FROM MEMORY) $PIN
QDST1M($DS)
QBRY1F($BR)
QDPB1F($DP)
BSAZ1F(Sel.Assoc.
Reg.)
1F Frame(Assoc.Reg.$_{0-35}$/
ZD0$_{36-59}$)
QZBC1F($BC)

Fig. 10.3.4.4 Store Associative Memory Zero (SAZ), Timing Diagram

10.3.4.5 Effective Address to Base$_n$(EABn) (See timing diagram, Fig. 10.3.4.5)

The EABn instruction involves strobing the new effective address in the BS-adder into the selected internal ABR. If an internal base must be added the segment tag register contains the designation of the base that is sent, (if internal) to the BS-adder.

Receiving $BORY generates $\$B_{13}$, sets the GBO flip-flop, inhibits BAPIIF(append for ESTR and OSTR/Sel. Int) continues to allow BAPZ1F (append to ESTR and OSTR/Sel ZBRI) and enables control points BEBP0I (ZI,4,7,8/Sel Int) and BBSB0F(¢BS_{0-17}/ABR_{0-17}). In addition, if the Control Frame is in the PA cycle of the EABn instruction with bit 29 of the instruction word = "1," or an ITB modifier was received, BIIB0F(¢ZBRI/toCF) is enabled indicating that an internal base must be added before strobing the selected internal ABR with the new effective address. Decoding of $ZI_{4,7,8}$/Sel Int. determines the BIN_OF(¢$INT_) level that controls $\$ABR_{0-17}$ for the selected register. The output of the applicable segment tag register (OSTR or ESTR) for the EABn instruction as determined by IC or $\overline{IC}$ selects the internal base to be sent to the BS-adder via the ZBRI/to CF switch. The data from the BS-adder is settled when $\$ABR_{0-17}$ is generated by the second $\$B_{13}$ from the $B chain; QIBD1F recirculating to generate the second $\$B_{13}$. Strobe $\$ABR_{0-17}$ is only generated if the Master = "1" or lock bit ($ABRn_{22}$) = "0" conventions are met and occurs approximately 180 nanoseconds after the first $\$B_{13}$ generated by $BORY. As a result of the second $\$B_{13}$, $BC is generated, the GBO flip-flop is reset and the enables to all the applicable control points are removed.

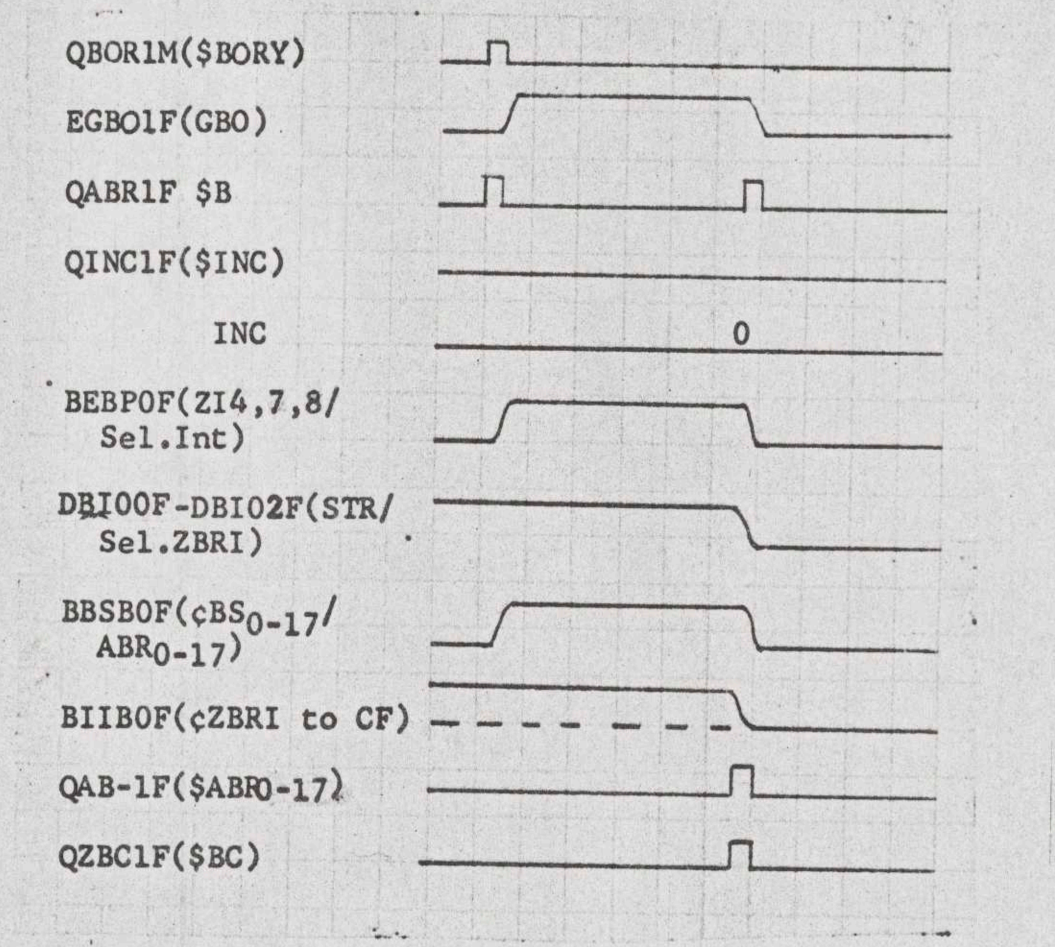

QBOR1M($BORY)
EGBO1F(GBO)
QABR1F $B
QINC1F($INC)
INC
0
BEBPOF(ZI4,7,8/ Sel.Int)
DBI00F-DBI02F(STR/ Sel.ZBRI)
BBSBOF(¢BS$_{0-17}$/ ABR$_{0-17}$)
BIIBOF(¢ZBRI to CF)
QAB-1F($ABR0-17)
QZBC1F($BC)

Fig. 10.3.4.5 Effective Address to Base n (EABn), Timing Diagram

10.3.4.6 Effective Address to Pair$_n$ (EAPn) (See timing diagram Fig. 10.3.4.6)

The EAPn instruction involves strobing the effective address into the designated internal ABR and strobing the pointer from the TBR into the linked external ABR designated by bits 18-20 of the internal ABR. If the ABR designated by the EAPn instruction contains an external base, then the pointer from the TBR is strobed into the same base as designated by the EAPn instruction.

Strobe $BORY initiates base frame operations by setting the GBO flip-flop, turning on control points BEBPOF(¢ZI$_{4,7,8}$/Sel Int), BBSBOF(¢BS$_{0-17}$/ABR$_{0-17}$), enabling the $\$B_{13}$ input to the increment counter and (if in the PA cycle of the EAPn instruction with bit 29 of the instruction word = "1" or an ITB modifier received) BIIBOF(¢ZBRI/to CF). Bits 4,7,8 of ZI are decoded to pick the designated register to be strobed. When an EAPn instruction is detected, the ¢$INT_ level(BINOOF) which controls the strobe selection is forced on whether the designated base is internal or external. The segment tag for the EAPn instruction is decoded from OSTR or ESTR as determined by IC or $\overline{IC}$ to select the internal base that is sent to the BS-adder in the Control Frame.

By the time that QAB_1F($\$ABR_{0-17}$) is generated as a result of the second $\$B_{13}$ from the recirculating $B chain, the BS-adder input to the ABR designated by the EAPn instruction is stable. Strobe $\$ABR_{0-17}$ for the selected register is a function of the Master/Slave mode, lock base bit (BSL01F) restrictions, and an increment count of "0." The $\$B_{13}$ that generates $\$ABR_{0-17}$ also advances the increment counter to "1," terminates level ¢BS$_{0-17}$/ABR$_{0-17}$, and enables BTBBOF (¢TBR$_{0-17}$/ABR$_{0-17}$) in preparation for putting the pointer in the TBR into the linked external base designated by bits 18-20 of the selected internal base.

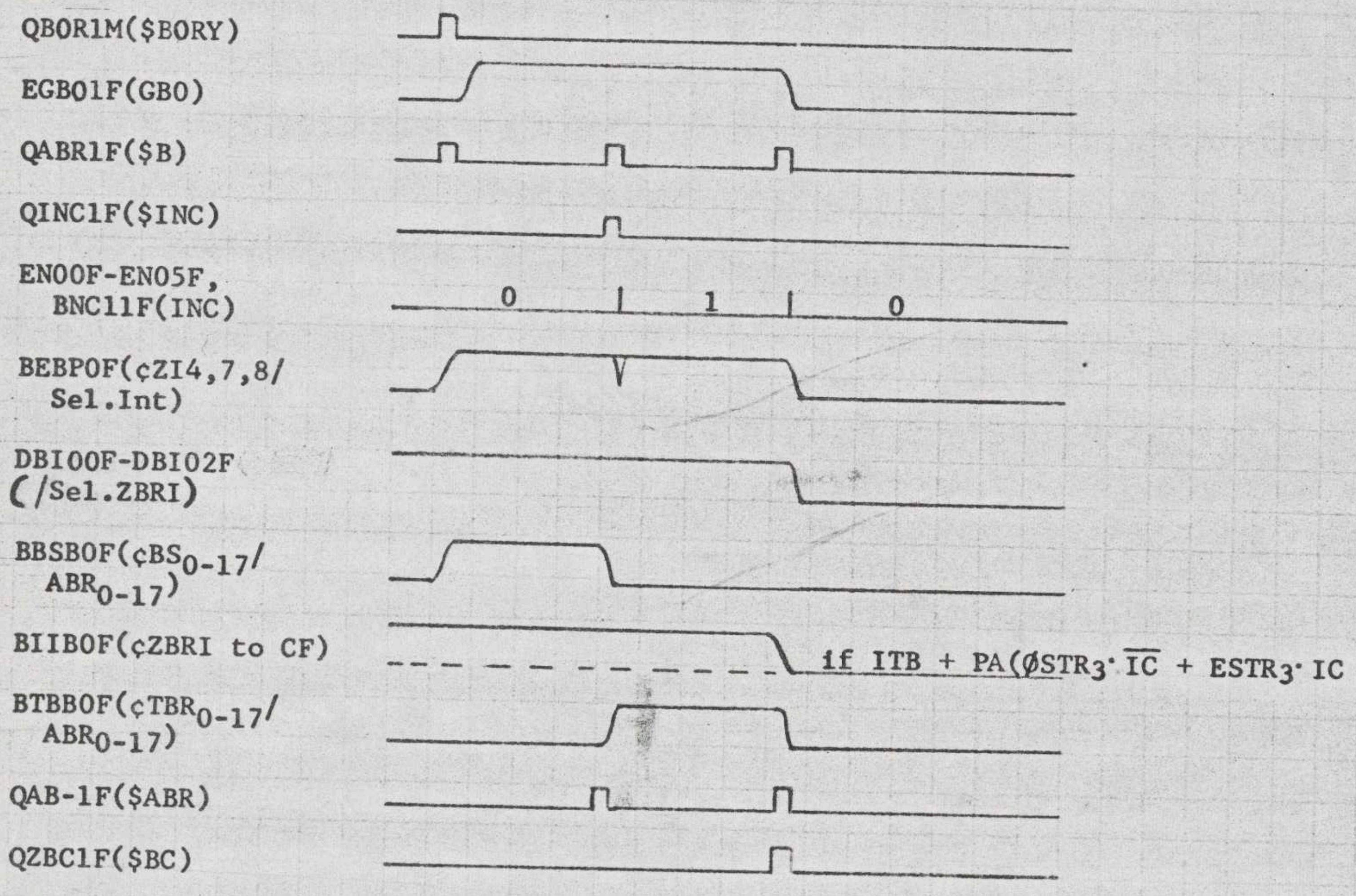

QB0R1M($B0RY)
EGB01F(GB0)
QABR1F($B)
QINC1F($INC)
EN00F-EN05F,
BNC11F(INC)
0
1
0
BEBP0F(¢ZI4,7,8/
Sel.Int)
DBI00F-DBI02F
(/Sel.ZBRI)
BBSB0F(¢BS_{0-17}/
ABR_{0-17})
BIIB0F(¢ZBRI to CF)
if ITB + PA(ØSTR$_3$·$\overline{IC}$ + ESTR$_3$·IC
BTBB0F(¢TBR_{0-17}/
ABR_{0-17})
QAB-1F($ABR)
QZBC1F($BC)

Fig. 10.3.4.6 Effective Address to Pair n (EAPn), Timing Diagram

10.3.4.6 - contd

Selection of the ABR to be strobed with the pointer from the TBR depends on whether the original ABR was internal or external. If the original ABR was internal the external base is decoded from bits 18-20 of the designated internal base that was dumped on the ZXBN bus, and the 5-wide NAND gate associated with the linked external ABR generates BRX-OF (¢$EX_). Level (¢$EX_) in addition to the decoded EAPn instruction, an Increment Count of "1," and BSL01F generates BAB_OF, which controls the strobe for the external register designated by bits 18-20 on the ZXBN bus. If the original ABR selected by the EAPn instruction was external, then the 3-wide gate associated with ¢$EX_ generation and the original ABR designated by the one of eight decoder, is enabled.

The receipt of the third $\$B_{13}$ from the $B loop generates $\$ABR_{0-17}$ for the proper ABR: the linked external ABR if the original ABR was internal; the EAPn designated ABR if the original ABR was external. Concurrent with the third $\$B_{13}$, the GBO flip-flop is reset, $BC is generated (assuming the Master/Slave lock base conventions were met and no fault was generated), all control point enables are removed, and the increment counter is reset to zero.

10.3.4.7 Load Base Register n (LBRN) (See timing diagram Fig. 10.3.4.7)

When performing an LBRn base operation the Base Frame in a sense interrupts a normal append operations sequence as soon as the Interface Frame has strobed the appended address for the "operand" into its Address Register and issued a $INT for a final address to memory. Strobe $BORY is received generating $\$B_{13}$ when a $INT is initiated by the Interface Frame and sets the GBO flip-flop. Setting the GBO flip-flop inhibits all control points concerned with Address Appending for operands, and enables BLSROF $(¢ZI_{6,7,8}/\text{Sel Int})$ and BMIBOF (ZDI_{0-23}/ABR_{0-23}). Decoding of $ZI_{6,7,8}$ picks the affected register through the generation of ¢$INT_. Level ¢$INT_ is forced on whether the selected $\overline{ABR}$ is internal or external and controls the selection of QAB-1F $(\$ABR_{0-17})$. Strobe QAC-1F$(\$ABR_{22})$ is a function of LBR·GBO·¢$INT·BMST1B (Master mode) and the next $\$B_{13}$. Similarily, strobe QAD-F$(\$ABR_{18-21,23})$ is a function of LBR·GBO·¢$INT·BSL-IF (Master = 1+ Lock = 0) and the next $\$B_{13}$.

Receipt of QSDA1M($DA) from the Memory generates $DA∆ and subsequently $\$B_{13}$, $\$ABR_{0-17}$, $\$ABR_{22}$, and $\$ABR_{18-21,22}$. Strobe $DA∆ is not delayed as long as for the ADBn or LDCF instructions since the extra delay in those cases was for the BS-adder in the Control Frame to stabilize. Strobe $BC is generated in the usual manner as previously explained, provided no lock base fault (¢ Set FLT) occurred. After $BC signals Base Frame complete, the append operations cycle is closed out in the normal manner upon the arrival of the next $C.

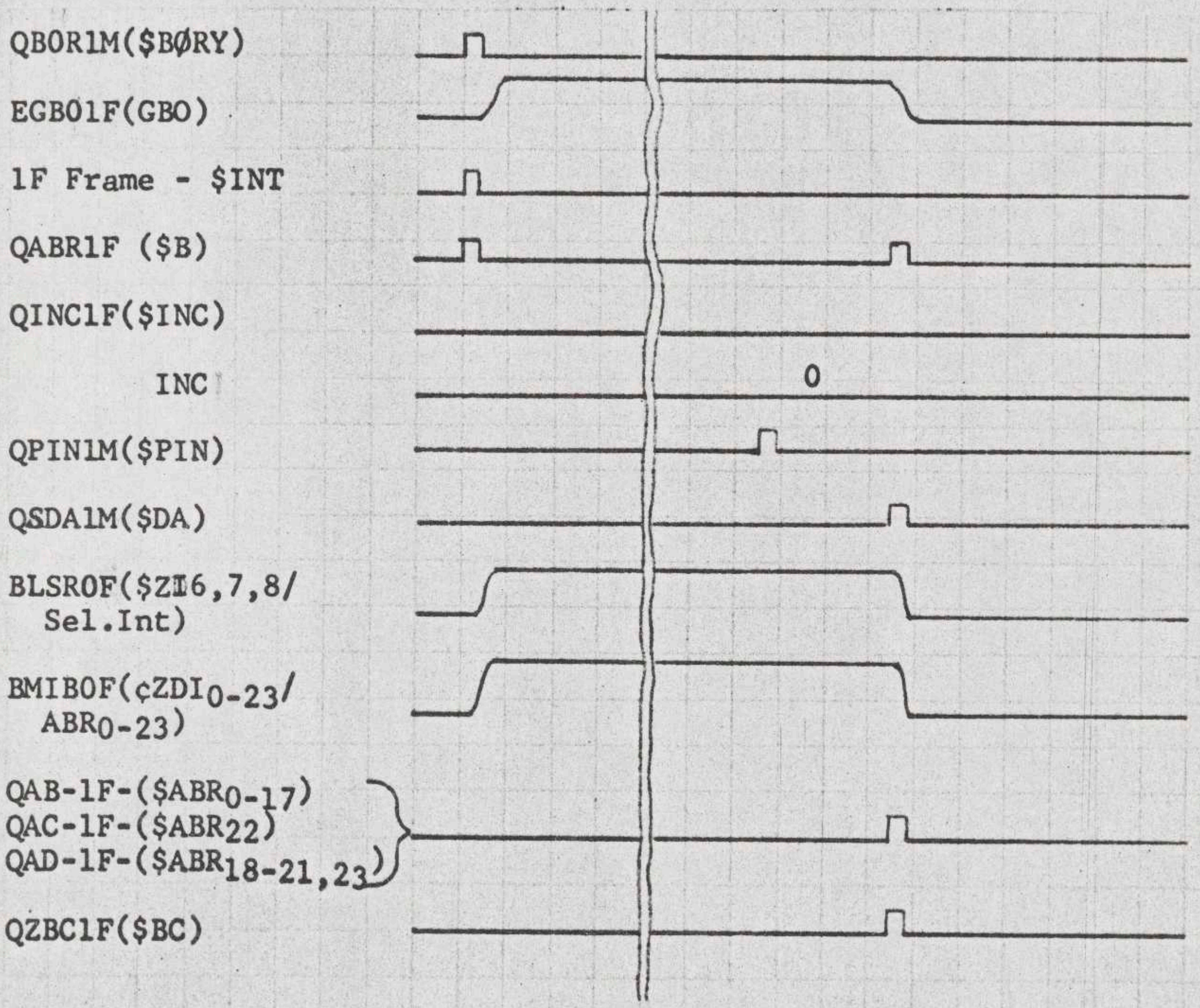

QB0R1M($B0RY)
EGB01F(GBO)
1F Frame - $INT
QABR1F ($B)
QINC1F($INC)
INC
0
QPIN1M($PIN)
QSDA1M($DA)
BLSR0F($ZI6,7,8/ Sel.Int)
BMIB0F(¢ZDI0-23/ ABR0-23)
QAB-1F-($ABR0-17)
QAC-1F-($ABR22)
QAD-1F-($ABR18-21,23)
QZBC1F($BC)

Fig. 10.3.4.7 Load Base Register n (LBRn), Timing Diagram

10.3.4.8 Load Base Registers (LDB) (See timing diagram Fig. 10.3.4.8)

The sequence of operations for the LDB instruction is almost indentical to that of the LBRn instruction, with the following exceptions: increment counter bits 3-5(2^2 - 2^0) are routed to the one of eight decoders to successively select the eight ABR's; no fault signal is sent to the Control Frame if an attempt is made to load the ABR's in Slave mode with ABR_{22} = "1," but the affected ABR will not be changed. Strobe \$BC is generated when the increment count reaches 7 and the next $\$B_{13}$ arrives (eight operations complete).

10.3.4.9 Load Descriptor Segment Base Register (LDBR) (See timing diagram Fig. 10.3.4.9)

When an LDBR instruction is recognized by the Control Frame, and the Processor is in Master mode, QBOR1M(\$BORY) is received from the Interface Frame generating $\$B_{11}$ and setting the GBO flip-flop. Data to load the DSBR comes over the ZDI lines from memory, and are strobed into the DSBR when \$DA from Memory generates $\$B_{11}$. Strobe \$BC is generated at $\$B_{13}$ time in the usual manner. No fault signal generation is a function of the LDBR instruction in the Base Frame load register fault detection circuitry, since the Processor fault logic makes a determination prior to the execution of the LDBR instruction. In conjunction with execution of the LDBR instruction a CAM·GBO (reset to Associative Memory) is generated.

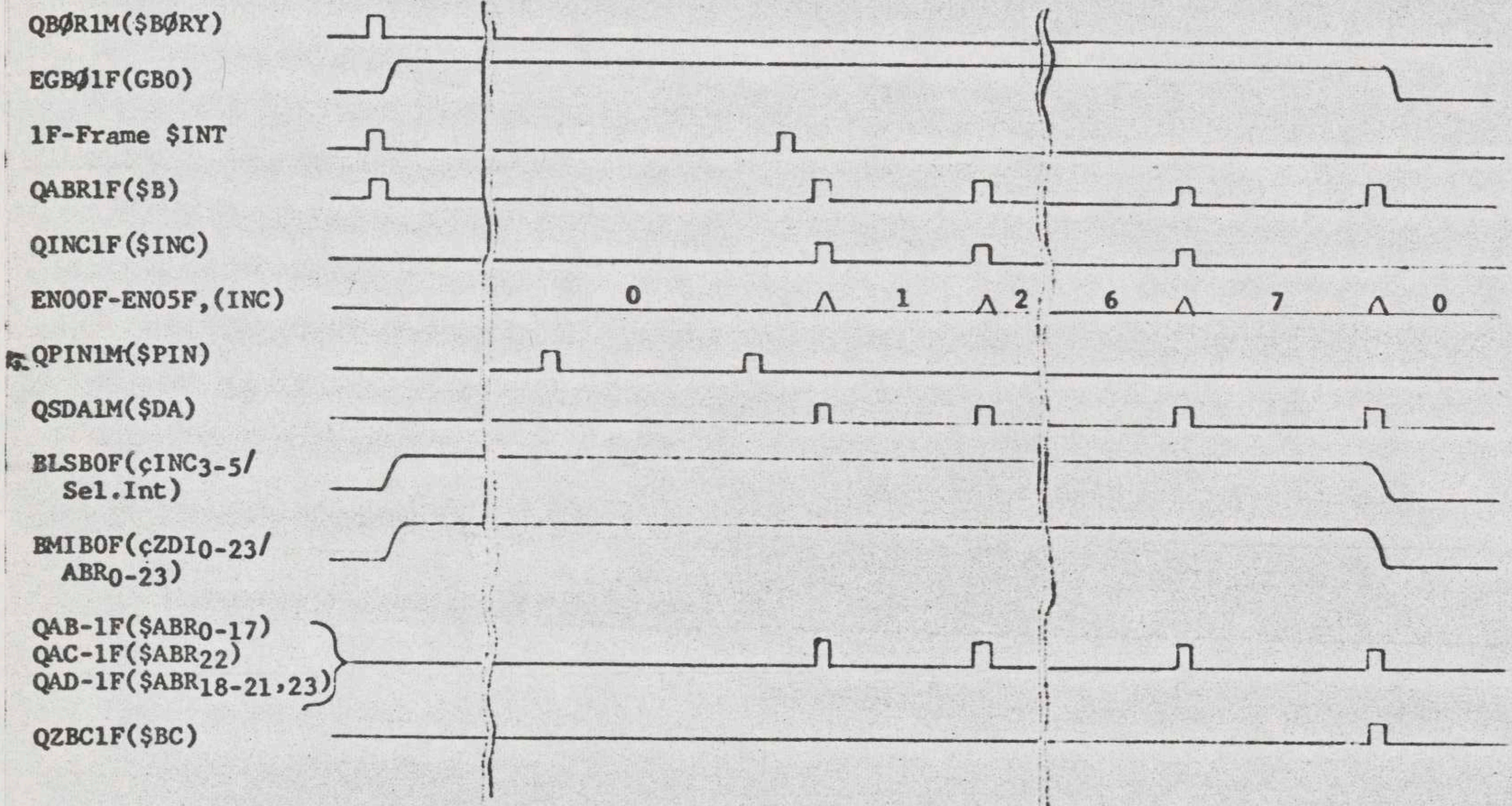

QBØR1M($BØRY)
EGBØ1F(GB0)
1F-Frame $INT
QABR1F($B)
QINC1F($INC)
EN00F-EN05F,(INC)
0
1
2
6
7
0
QPIN1M($PIN)
QSDA1M($DA)
BLSB0F(¢INC3-5/ Sel.Int)
BMIB0F(¢ZDI0-23/ ABR0-23)
QAB-1F($ABR0-17)
QAC-1F($ABR22)
QAD-1F($ABR18-21,23)
QZBC1F($BC)

Fig. 10.3.4.8 Load Base Registers (LDB), Timing Diagram

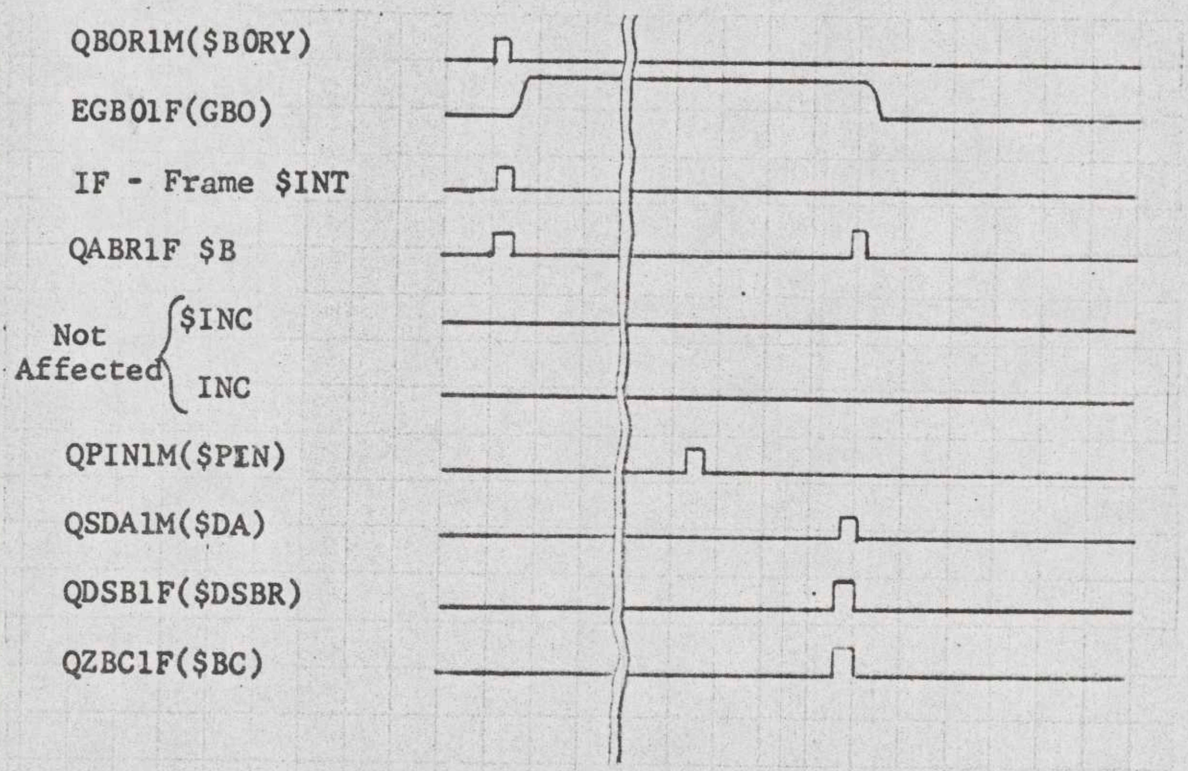

QBOR1M($BORY)
EGB01F(GBO)
IF - Frame $INT
QABR1F $B
Not Affected
$INC
INC
QPIN1M($PIN)
QSDA1M($DA)
QDSB1F($DSBR)
QZBC1F($BC)

Fig. 10.3.4.9 Load Descriptor Segment Base Register (LDBR), Timing Diagram

10.3.4.10 Load Control Field (LDCF) (See timing diagram Fig. 10.3.4.10)

For this instruction the effective address in the BS-adder in the Control Frame is interpreted as three 6-bit characters: character 1 contains the ABR designation, (BS_{0-2} for ABRn, BS_{3-5} for ABRm), character 2 the new information for ABRn, character 3 the new information for ABRm. Strobe \$BORY generates $\$B_{13}$ setting the GBO flip-flop resulting in control points BLDOOF(¢BS_{0-2}/Set Int, BS_{6-11}/$ABRn_{18-23}$) being turned on. When \$DA is received from the Memory signifying that data is on the lines to the Control Frame, $\$DA_{\Delta}$ is generated (delayed approximately 250 nanoseconds after \$DA to allow time for the BS-adder to settle) and generates $\$B_{13}$. Bits 0-2 of the BS-adder are then decoded to select the proper ABR to be loaded with new control field information, and enable ¢\$INT_ to the ABR strobe circuits. Strobe $\$B_{13}$ causes the generation of strobes for ABR_{0-17}, ABR_{22}, $ABR_{18-21,23}$ if the Processor is in Master mode or the selected base is not locked against change when the Processor is in Slave mode. Though strobes for the entire 24 bits are present, only the control field data(bits 18-23)are present.

The $\$B_{13}$ that generated $\$ABR_{0-23}$ (three strobes) advances the increment counter to "1" thereby removing BLDOOF and enabling BLDIOF (¢BS_{3-5}/Set Int, ¢BS_{12-17}/$ABRm_{18-23}$). When the increment count reaches "1" the internal $\$B_{13}$ loop is enabled by (LDCF)(INC=1) to generate another $\$B_{13}$; generating $\$ABR_{18-23}$ as before, but for the new ABRm specified by BS_{3-5} and for the new data from BS_{12-17}. This same $\$B_{13}$ generates \$BC and resets the GBO flip-flop. With this particular instruction if an attempt is made to load control fields with the ABR's designated locked against change then strobe QFLT1F(\$FLT) will be generated, one strobe for each ABR attempted, though only the first one is sufficient for the Processor fault logic.

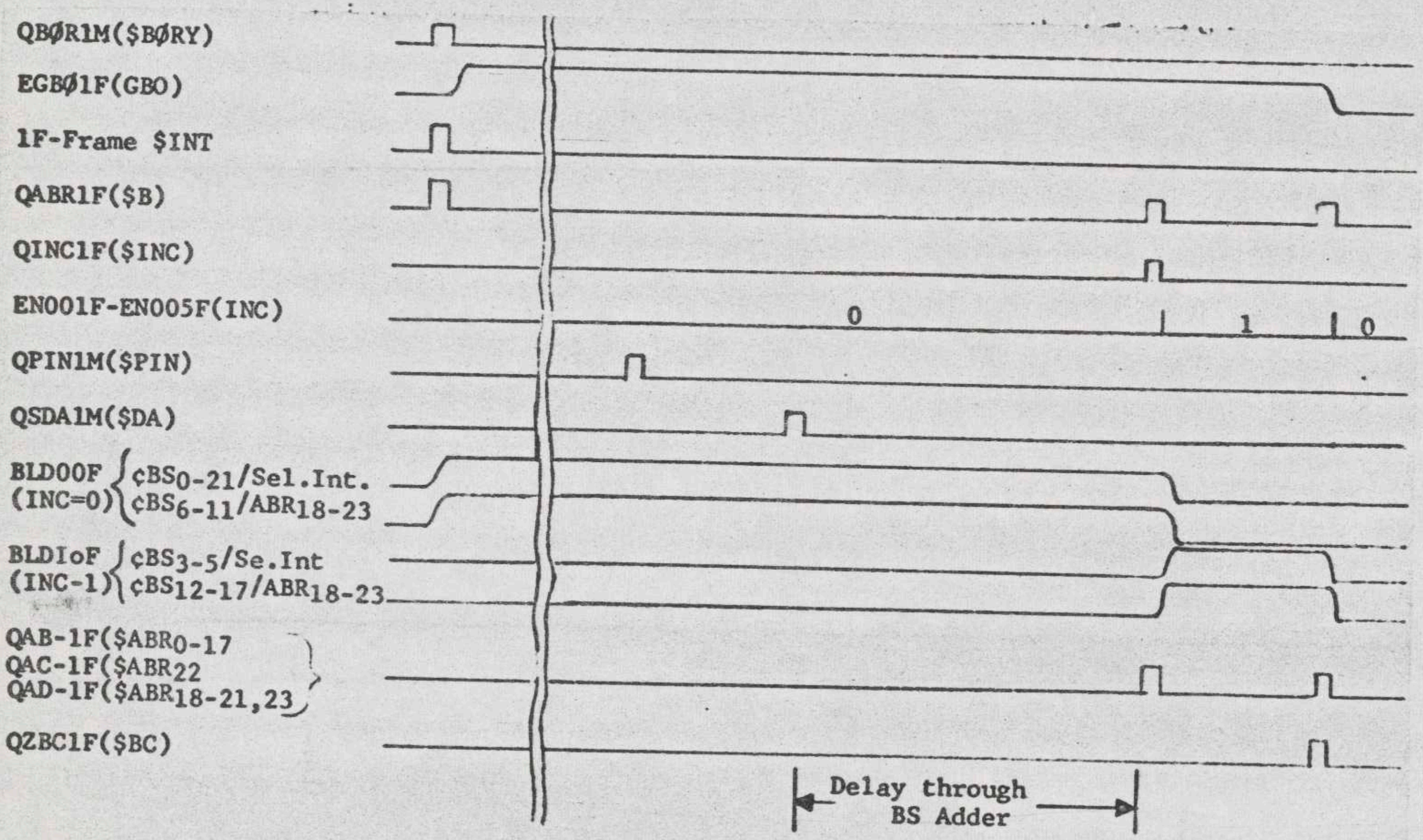

QBØR1M($BØRY)
EGBØ1F(GBO)
1F-Frame $INT
QABR1F($B)
QINC1F($INC)
EN001F-EN005F(INC)
0
1
0
QPIN1M($PIN)
QSDA1M($DA)
BLD00F (INC=0)
¢BS0-21/Sel.Int.
¢BS6-11/ABR18-23
BLDIoF (INC-1)
¢BS3-5/Se.Int
¢BS12-17/ABR18-23
QAB-1F($ABR0-17
QAC-1F($ABR22
QAD-1F($ABR18-21,23
QZBC1F($BC)
Delay through BS Adder

Fig. 10.3.4.10 Load Control Field (LDCF), Timing Diagram

10.4 PROCESSOR POWER ON/OFF SEQUENCING (See 43D163422 sheet 22)

(To be supplied at a later date.)

10.5 PROCESSOR FAULT LOGIC

10.5.1 GENERAL DESCRIPTION

The fault logic circuitry for the Processor detects the conditions which will result in one of 32 defined Processor faults and causes a trap to the memory cell (two consecutive memory locations; even - odd) in the Processor Fault Vectors corresponding to the particular fault. In performing this function the fault logic circuitry does the following:

1) Anticipates that a fault condition may occur and sets a number of preliminary flags or flip-flops to help define the fault should it occur.

2) Detects the conditions which will result in a fault.

3) Depending on the type of fault detected and fault priority, waits until the optimum time before starting any actions.

4) When conditions are suitable, initiates a "snapshot" sequence to save the status of the Processor and the conditions that caused the fault. The term "snapshot" in this case indicates exactly what takes place; a picture of the Processor with the conditions which caused the fault safestored in the Processor internal registers.

   As a result of the "snapshot" sequence, Processor flags and registers are saved awaiting an SCU instruction as the first instruction of the fault vector pair for the particular fault detected. The "snapshot" saves the following in the registers indicated:

   a) Temporary Base register saved in the Data out register, bits 0-17.

   b) Appending Unit status flags saved in the Data out register bits 18-35 as follows (the bits indicate control of a register or that a flag is set):

10.5.1 - contd

b) Appending Unit status flags saved in the Data out register bits 18-35 as follows (the bits indicate contents of a register or that a flag is set):

Bit Position	Name
18-21	$OSTR_{0-3}$ register
22-25	$ESTR_{0-3}$ register
26	ITS tag
27	ITB tag
28	not used ("0")
29	PEO flag
30	ITR-indirect tally not equal to Tally Runout Indicator
31	not used ("0")
32	not used ("0")
33	DSPTW fetch
34	SDW fetch
35	PTW fetch

c) Computed address existing during the last address preparation cycle saved in the Data out register, bits 36-53 (this is the effective address).

d) Control Frame status flags saved in the Data out register, bits 54 - 71 as follows (the bits indicate contents of a register or that a flag is set):

Bit Position	Name
54	PI flag
55	PNO flag
56	not used ("0")
57	XDE flag

10.5.1 - contd

Bit Position	Name
58	XDO flag
59	IC flag ("0" is even instruction. "1" is odd instruction).
60	MASF flag
61	EA level ("1" indicated operand fetch; "0" indicates indirect fetch)
62	MSY flag ("1" indicates Master mode; "0" indicates Slave mode)
63	PA flag
64	PZ flag
65	PT flag
66 - 71	CT register

e) The ICT address of the faulting instruction is saved in the ICTA register.

f) The contents of the Y- and C-registers are saved in the I-register and reflect any address modification for operand address preparation that had taken place prior to the "snapshot" sequence.

5) Initialize the Control, Base and Interface Frames with a WF level and the Operations Unit with a strobe $ABORT. Both signals being generated as a result of the "snapshot" sequence.

6) Initiate the actions to enable the Control Unit to "brute-force" and XED instruction into both the Y- and C-registers with an address corresponding to the fault detected and processed by the fault logic.

7) Wait until the first of the XEDed instructions if an SCU, or until the XED instruction are executed without a fault, then reset the fault logic for detecting a trouble fault.

10.5.1.1 Fault Descriptions

The 32 Processor faults fall into distinct groups for recognition and priority of service. These groups are designated Group I through Group VI, with the faults in Group I having the highest priority and Group VI the lowest priority. Within each group, the faults are mutually exclusive (no two can exist at one time) with the exception of Parity (which takes priority over Group III faults) and Group VI faults. For faults in Group VI, Shutdown has the highest priority followed by Timer Runout, and then Connect. External interrupts even though they are considered as Group VI faults, have the lowest priority of service. Interrupts will not be honored if any fault in Groups I through Group VI is detected before the Processor enters a WC cycle for an execute interrupt command response from the Memory Controller.

A summary of the various processor recognized faults is listed in Table 10.5.1.1.

10.5.1.2 Fault Detection and Recognition

Each of the 32 recognized Processor faults results in the same processor response, but causes a different fault vector address depending on the fault being serviced. However, the detection of the fault, and the subsequent recognition of the fault for service, are distinctly different for each fault condition. Table 10.5.1.2 lists the different groups of faults and the conditions under which the faults in each group are detected and recognized.

Table 10.5.1.1 Summary of Recognized Processor Faults

Fault Group	Fault Code	Fault Name	Fault Category	Fault Description
I	74	Startup	Manual	Processor initialized after POWER ON pushbutton depressed resulting in a DIS state. Then fetch two instructions from Startup fault vector.
	66	Execute	Manual	EXECUTE pushbutton depressed results in fetching two instructions from the Execute fault vector, or loading the Maintenance Panel INSTRUCTION switch setting into both instruction registers.
II	76	Trouble	Hardware	A Trouble fault occurs if a fault is encountered during a "brute forced" XED instruction for a fault or external interrupt or during the execution of the SCU instruction as the first instruction of the fault vector designated by the XED address.
	72	Operation Not Complete	Hardware	Any one of the following: (1) Addressed Memory Controller not connected to Processor, (2) Memory Controller does not respond to Processor, (3) Processor has not generated memory access or had a direct operand for a 1-2 milliseconds and is not in a DIS state, (4) RAR cycle is closed out by Memory Controller without receiving data from Processor, (5) Memory does not acknowledge data from Processor for DP store or RAR cycle, (6) Programming rules violated leading to undefined operation.

Table 10.5.1.1 Summary of Recognized Processor Faults (contd)

Fault Group	Fault Code	Fault Name	Fault Category	Fault Description
III	64	Divide Check	Program	Division cannot be complete as defined by the instruction.
	62	Overflow	Instruction Program	Any arithmetic overflow that is not masked by the Overflow Mask indicator.
IV	30	Parity	Hardware	Any one of the following parity errors that are not masked by the Parity Mask indicator (except parity on SDWs or PTWs which are independent of the Parity Mask): (1) Single-precision reads; (2) Double-precision reads, either word; (3) Read-alter-rewrite cycles with parity on read, the rewrite cycle is completed with correct parity (4) Instruction or indirect word (5) Operand for Operations Unit is executed with bad parity then the Parity fault is recognized.
	32	Illegal Memory Command	Instruction Program or Hardware	Processor has issued a connect command to a masked Memory Controller channel.
	70	Lockup	Instruction Program or Hardware	No recognition of external interrupts for 1-2 milliseconds and the Processor is not in a DIS state. Lockup is disabled on "repeat" instructions.

Table 10.5.1.1 Summary of Recognized Processor Faults (contd)

Fault Group	Fault Code	Fault Name	Fault Category	Fault Description
V	26	Illegal Descriptor	Operating System	Bits 30-35 of the last SDW or PTW are undefined.
	24	Illegal Procedure	Instruction or Program	Generally programming violations: (1) GE-645 privileged instructions in slave - LDBR, SDBR, SAM, ZAM, CAM, SCU, RCU, LACL (2) GE-645 semi-privileged instructions in slave and base is locked - EAPn, EABn, TSBn, LDCF, ADBn, LBRn (3) Undefined operation codes including the all zero code (4) Out of bounds address or pointer (5) Access restrictions on the segment or page Processor is attempting to access.
	22	GE-635 Compatiability	Instruction or Program	Traps instructions in the GE-635 repertoire not included in the GE-645 - LBAR, SBAR
	60	GE-635/645 Compatability	Instruction or Program	Attempted to execute GE-635/645 privileged instructions in Slave instead of Master mode - SMIC, RMCM, SMCM, CIOC, LDT, DIS, TSS
	2,10, 12,16	MME1-4	Instruction or Program	Programmed fault defined by instruction operation code and address.
	4	Derail	Instruction or Program	Programmed fault defined by instruction operation code.
	20, 34, 36	Fault Tag 1-3	Instruction or Program	Programmed fault in instruction or indirect word tag for an IT modification, causes an immediate fault, F1 is for GE-635 programs; F2, F3 are for GE-645 programs.

Table 10.5.1.1 Summary of Recognized Processor Faults (contd)

Fault Group	Fault Code	Fault Name	Fault Category	Fault Description
V	40,42, 44,46, 50,52, 54,56	Directed Faults 0-7	Operating System	Defined by bits 30-31 of an SDW or PTW as to the type of Directed Fault, for example, missing page from a data segment.
VI	0	Shutdown	Manual	Power will be turned off to the Processor approximately 1 millisecond after the POWER OFF pushbutton is depressed.
	6	Timer Runout	Hardware	This fault is generated when the contents of the Timer Register reaches zero and the Processor is in Slave mode. If the Processor is in Master mode the Timer Runout fault is not recognized until a return to the Slave mode.
	14	Connect	Instruction or Program	A Connect pulse from another active device is received via the Memory Controller
The fault vector address is received from the Memory Controller in response to an execute command sent from the Processor in response to an external interrupt		External Interrupts	Hardware	A device has set an interrupt cell in the Memory Controller resulting in an interrupt present request received at a Processor port.

10.5.1.2 - contd

Detection means exactly that; a fault condition has been detected. Recognition occurs when the fault logic initiates a "snapshot" sequence leading to the "brute forced" XED instruction with the proper fault address.

Table 10.5.1.2 Groups I through VI Fault Detection and Recognition

Fault Group	When Detected	When Recognized
I	Manually initiated from Processor Maintenance Panel	The Processor is initialized and faults unconditionally upon detection.
II	The Trouble fault is detected any time that a fault occurs during the formulation and execution of an XED instruction for a fault or external interrupt, or the execution of the SCU instruction which is the first instruction of the vector designated by the XED instruction.	As soon as the Trouble fault is detected the new XED instruction is formed with the address of the Trouble fault vector. A second "snapshot" sequence is not performed.
	Operation not Complete is detected if no strobe $R is generated by the Control Unit for 1-2 milliseconds; the time interval being determined by a 1-kc oscillator in the Control Unit. No $R from the Control Unit implies any one of the six conditions listed in Table 10.5.1.1.	As soon as recognized, a "snapshot" and fault sequence is initiated with a fault address corresponding to the Operation not Complete fault vector.

Table 10.5.1.2 - contd

Fault Group	When Detected	When Recognized
III	Divide Check and Overflow are detected by the Operations Unit during the execution of an instruction	As soon as recognized and the Memory Controller has responded to the last access request.
IV	The faults in this group (Parity, Illegal Memory Command and Lockup) are detected after the address preparation cycle that caused the fault. Parity and Illegal Memory Command are detected when an illegal action code is received from the Memory Controller about 0-400 nanoseconds after the Memory Controller has acknowledged the last access request. The Lockup fault is detected from 1-2 milliseconds after an external interrupt request is present and has not been acknowledged.	These faults are recognized as soon as all previous operations have been executed and the Memory Controller has acknowledged the last access request.

Table 10.5.1.2 - contd

Fault Group	When Detected	When Recognized
V	These faults are detected during the address preparation cycle in the Control Unit that generates the fault.	Recognition of these faults takes place after the completion of any instructions currently being executed. Any access to memory for an indirect word or operand for the current instruction being prepared is inhibited.
VI	These faults are detected in the beginning of any address preparation cycle for an operand or indirect word. (Approximately the first 225 nanoseconds into the cycle).	These faults are recognized as soon as the Processor has completed all pending operations provided the Processor is not in Master mode and the interrupt inhibit bit of the current instruction being prepared is not set. Timer Runout is not recognized if the Processor is in Master mode, until a return to Slave mode.

10.5.2 FAULT DETECTION

10.5.2.1 Group I Faults (See 43D163422 sheet 21)

Since the faults in Group I are manually initiated, detection is a matter of converting the manually initiated contact closures to a logic level indicating the presence of the fault.

Startup - When the POWER ON switch is depressed, the power sequencing logic for applying a-c power to all d-c power supplies, initializes the processor to a DIS state by generating a STOP signal for the fault sequence initiation logic in the Base Frame. During the initialize phase, the PSUF flip-flop is set by the sequence initialize logic and remains set until the Processor is initialized and a DIS instruction is forced into the Y- and C-registers. During the PA cycle for the DIS instruction PSUF still set forces a fault sequence for a Startup fault. To recap slghtly; detection of the Startup fault is accomplished when the PSUF flip-flop is set by the initialize logic for power on sequencing.

Execute - Detection of the Execute Fault is discussed in detail in paragraph 9.3.12.1.

10.5.2.2 Group II Faults

Trouble (see 43D163422 sheet 17) - The Trouble fault is recognized during the address preparation and execution of an XED instruction for a fault, or during the execution of an SCU instruction as the first instruction of the fault vector pair. This time interval is determined by the set and reset conditions for the WFTB flip-flop. The WFTB flip-flop is set by the strobe $C that strobes the "brute forced" XED instruction for the particular fault detected into the Y- and C-instruction registers. Termination of the interval is accomplished by the XEDed instruction in the fault vectors being successfully loaded into the Y- and C-register (no illegal actions from memory). If an SCU instruction is the first instruction of the fault vector, WFTB is not reset until the SCU instruction is successfully executed.

10.5.2.2 - contd

When WFTB is set, any illegal action from memory, Operation not Complete fault or Illegal Procedure fault sets the TRBL flip-flop initiating a sequence for the Trouble fault. The TRBL flip-flop is reset when a $C is generated to strobe the XED instruction designating the Trouble fault vector into the Y- and C-registers.

Operation Not Complete (see 43D163422 sheet 21) - This fault is defined by the state of the MSFF flip-flop which determines the start of the 1-2 millisecond interval to detect Operation not Complete:

1) MSFF is set by a 1-kilocycle pulse from the Control Unit.

2) The PONC flip-flop is set with the next 1-kilocycle pulse from the Control Frame unless a $R is issued for an interrupt and MSFF is reset.

3) PONC being set initiates an Operation not Complete fault.

4) PONC is reset as soon as a $C is issued that results in the XED instruction for a fault being strobed into the Y- and C-registers.

10.5.2.3 Group III Faults

The Group III faults are detected by the Operations Unit during the process of trying to execute an instruction. As a result, the Fault Logic receives one of two levels (FOLF, DVDK) from the Control Unit and generates a GP3 level (BP031F) to start a fault sequence. The flip-flops for holding the FOLF and DVDK levels are contained in the Control Unit. These flip-flops are reset when the XED instruction for the particular fault is strobed into the Y- and C-registers.

10.5.2.4 Group IV Faults

Parity (see 43D163422 sheets 19 and 20). Detection of a parity fault is more difficult than other Processor faults due to the many possible conditions that can cause the fault. In addition, the illegal action strobe from the Memory Controller for a parity error comes approximately 400 nanoseconds after the data is on the lines to the Processor. This means that the Control Unit can be approximately 250 nanoseconds into the next address preparation cycle for an instruction, indirect word or operand before any indication that a parity error has occurred. For these reasons, two-time intervals designated as "windows" are set up to determine where the parity error has occurred, should one occur. Detection of the parity error takes place during the second "window" sequence.

The first window sequence is determined by any one of the following conditions being true:

1) RFO
2) RFI
3) $WI_2 \cdot \overline{RCU}$
4) $\overline{PI\ (XDE \cdot SPI)}$

RFO (Request for Operand) is a flag set whenever a $INT to memory is issued for an operand. Resetting of RFO is accomplished by a $PIN signifying that the Memory Controller has responded to the $INT.

RFI (Request for Instruction) is a flag set whenever a $INT to memory is issued for a nonbuffer filling instruction (indirect word SDW, PTW) that is, instructions that are not going into the I-register. Other cases are where an EX or EXI function is generated, an XED instruction in the Y-register is being executed, or a double PI cycle is being executed because of an RCU instruction or access violation. These other cases are defined by the term PI (XDE + SPI).

$\underline{WI_2 \cdot \overline{RCU}}$ is a term that represents cases where a PI cycle is started to fill the I-register. This includes fetching of the SDW or PTW before the final fetch for the instruction pair.

10.5.2.4 - contd

PI($\overline{XDE}\cdot\overline{SPI}$) - is a term which represents the case of parity occurring within a PI cycle as distinguished from the case of parity as the result of a PI cycle.

The first window sequence is necessary to define the different possible cases where parity can occur. These different cases must be defined so certain areas of the Processor can be allowed to proceed or be stopped prior to initiating the fault sequence.

The second window sequence is set to detect the parity fault should one occur. This sequence is defined by any one of the following conditions being true:

1) PNO
2) PNI
3) PNB
4) BFPI (PI($\overline{XDE}\cdot\overline{SPI}$))

PNO (Parity for Operand) is a flag set when the memory has acknowledged an interrupt and the RFO flag was set: ($PIN·RFO.) Reset of PNO occurs with a strobe illegal action ($IAL)-delayed 103 nanoseconds-as a result of a parity error.

PNI (Parity for Instructions) is the flag set when RFI was set at the time that an interrupt was acknowledged by the memory: ($PIN·RFI.) Reset is the same as PNO.

PNB (Parity as a result of buffer filling instructions) is set whenever $WI_2\cdot\overline{RCU}$ is true at $PIN time. Reset is the same as PNO and PNI.

$\overline{BFPI}$ this level when true indicates that any parity error should it occur, will take place within a PI cycle started to fill the I-register.

10.5.2.4 - contd

Detection of parity errors is possible now that the second window sequence is entered. At this time look at sheet 20 of the logic diagrams and the inputs to the PEO, PEI, PFI, and PWP flip-flops. Any one of these flip-flops becoming set will initiate a fault sequence resulting in a Parity fault vector address. Setting of each flip-flop is a function of a $IAL (Illegal Action) from the Interface Frame and the PNO, PNI, PNB, BFPI levels in the second window sequence. The way the second window sequence is defined, the only $IAL that can occur is for a parity error. One other flag is set to further define parity; SEP. The SEP flag is set if PEO was set at the time the Operations Unit started a GS cycle for executing a new operand, after the operand that caused the parity fault was completed.

As stated before, the fault sequence initiation for the Parity fault varies depending on what kind of Parity fault is detected. Paragraph 10.5.3 covers fault sequence initiation.

Illegal Memory Command (see 43D163422 sheet 20) - The Illegal Memory Command fault is generated any time there is an illegal action strobe received from the Interface Frame as a result of a CIOC instruction to a channel masked off in the Memory Controller.

Lockup (see 43D163422 sheet 21) - A Lockup fault is recognized if no $R is issued (implying no $INT to memory) for 1-2 milliseconds and an external interrupt or Group VI fault is present. When an external interrupt or Group VI fault is present:

1) The IRP flip-flop is set on the next $C and this indicates an interrupt is present.

2) The next 1-kilocycle pulse from the Control Unit sets the LSFF flip-flop unless IRP is reset before the 1-kilocycle pulse arrives. Measuring the time interval for detecting Lockup begins when LSFF is set.

10.5.2.4 - contd

3) Unless LSFF and IRP are reset before the next 1-kilocycle pulse from the Control Unit arrives, PLUF will set and a Lockup fault routine will be initiated. Resetting of IRP and LSFF is dependent on a $C resetting IRP; implying that no interrupt is present that is not recognized.

10.5.2.5 Group V Faults

Illegal Descriptor (see 43D163422 sheet 14) - Detection of this fault is checked by the Interface Frame whenever an SDW or PTW is brought in from memory. The Interface Frame checks the descriptor field for an illegal descriptor code, and if one is detected, sends a strobe $ICF to the fault logic, setting the FID flip-flop. FID set results in a fault sequence being initiated.

Illegal Procedure (43D163422 sheets 10, 11, 12, 14) - The Illegal Procedure fault is generated primarily because of programming violations detected during the execution of a program. There are five cases to be considered:

1) GE-645 privileged instructions trying to be executed in Slave mode.

2) GE-645 semi-privileged instructions trying to be executed in Slave mode when the Address Base Register designated by the instruction is locked against change.

3) Undefined operation codes, including the all zero operation code.

4) A Pointer or an address out of bounds.

5) Access restrictions on the location being accessed. These restrictions are defined by the descriptor field of the SDW or PTW used to develop the address for the location to be accessed.

10.5.2.5 - contd

Case 1: The PIF flip-flop (sheet 12) is set whenever any GE-645 instructions that are privilege instructions, are decoded and the Processor is not in Master mode. PIF is set by the $C (delayed approximately 288 nanoseconds) that started the PA cycle for the instruction, and reset when the wait for trouble (WFTB) flip-flop is reset.

Case 2: Lock base faults are detected in the Base Frame any time that an Address Base Register is about to be changed, the base register has its "lock-base bit" set, and the Processor is not in Master mode. When detected, $LBF is received by the fault logic and sets the LBF flip-flop (sheet 14). LBF set results in a level to start a fault sequence. Reset of LBF is concurrent with the reset of the wait for trouble flip-flop.

Case 3: The illegal operation codes are detected during the PA cycle for the particular instruction. As with other program type faults, the $C that started the PA cycle during which the fault was detected, is delayed approximately 288 nanoseconds to give the instruction decodes time to settle before setting any flags. Should a faulty operation code be detected, the FOC flip-flop (sheet 11) is set resulting in a fault sequence being started.

Case 4: Any out of bounds address for a pointer indirect word or operand is detected by the boundary comparator in the Interface Frame. As a result a strobe $OBF is received by the fault logic to set the OOB flip-flop (sheet 14) and initiate a fault.

Case 5: Access violations for access restrictions write-permit, or class conventions are detected by the Interface Frame and a strobe $AVF is sent to the fault logic. This results in the AVF flip-flop (sheet 14) being set and initiating a fault sequence.

10.5.2.5 - contd

GE-635 Compatibility, GE-635/645 Compatibility, Master mode entry 1-4, Derail, Fault Tag 1-3 (See 43D163422 sheet 12). All these faults are classified as program faults and are lumped together as far as the decodes are concerned. The net result is that the PRF flip-flop is set at $C∆ time if a fault is detected. However, as far as defining the different faults are concerned, the fault addresses are different. This is accomplished by developing the different fault addresses required, using specific decode levels from the program faults decode networks that is, CF1, CF2, FLT, MMD, etc. The combination of the various decode levels and the PRF flip-flop initiate the fault sequence and develop the correct fault address.

Directed Faults 0-7 (see 43D163422 sheet 14) - The Directed Faults are detected by the Interface Frame at the same time that the descriptor field of an SDW or PTW is checked for an Illegal Descriptor fault. If a directed fault is detected then a $DF is generated and sent to the Base Frame fault logic to set the DF flip-flop. Determination of which Directed fault is detected, is developed during the formulation of the fault address encode. The Interface Frame sends the code for the particular fault that was detected. Initiation of the fault sequence is started by the DF flip-flop "setting."

10.5.2.6 Group VI Faults (see 43D163422 sheet 16) - Group VI faults (Shutdown, Timer Runout and Connect) are honored anytime a higher priority fault has not occurred. The setting of the PSHUF, PTROF, or PCON flip-flops will eventually result in the GP6 flip-flop being set with a strobe $IPR, and initiating a fault sequence. The reset input logic to the PSHUF, PTROF and PCON flip-flops serves two purposes: prevents any one of the individual flip-flops from being reset until a fault is generated for the particular fault detected; results in establishing a priority for the fault address encoded during the formulation of the XED instruction for the Group VI fault detected.

10.5.2.6 - contd

Group VI fault priority is determined by the fault code address which is generated, rather than the priority logic which leads to starting the fault sequence.

There is a special case for Group VI faults and terminate conditions for "repeat" instructions. Before a terminate condition is serviced, the Processor must not have started another fault sequence for Group VI faults. Assumming this condition is true; the GP6 flip-flop will be reset and the terminate condition serviced by the Processor. If any action had started for servicing a Group VI fault, the GP6 flip-flop will be reset for terminating a "repeat" instruction providing: the GSF flip-flop is set indicating the Interface Frame is stopped; or the CSR flip-flop is set indicating the Control Frame is stopped; or SSI from the Interface Frame is present indicating that a $INT has been issued and it is safe to start servicing a terminate condition. Terminate must be serviced prior to the beginning sequence for a fault or at the end of the sequence, but before the fault is generated. This is so the state of the various sections of Processor logic concerned is known. If a higher priority fault occurs at the same time as a Group VI fault and terminate condition, the GP6 flip-flop is reset by the term RS6 and the higher priority fault is serviced. However, the term TERM is included in the RS6 equation and would normally reset GP6 for a terminate condition. To prevent this, GP6 is inhibited from being reset by RS6 for a terminate condition. For the repeat terminate condition, signals GSF + CSR + SSI collector reset GP6.

10.5.3 FAULT SEQUENCE INITIATION

A fault sequence to service a detected fault condition is initiated on a priority basis depending on the fault group and type of fault in that group. The logic string for priority determination is shown at the top of 43D163422 sheet 14, and the inputs to the SABT level generation on sheet 15. The string of logic on sheet 14 is primarily to generate the reset terms

---

10.5.3 - contd

(RS3, RS41, etc.) for lower priority faults, while the actual priority logic for initiating the fault sequence is shown on sheet 15. Fault priority for Groups I through V is determined by the input logic configuration to the SABT and SFF level generation network and is of the "brute force" type variety; the higher priority overriding the lower priority faults to start a fault sequence and resetting any lower priority faults. Group VI faults are not reset by higher priority faults being serviced, and will eventually initiate a fault sequence. Resetting of the Group VI fault circuitry is entirely dependent on the fault being serviced.

10.5.3.1 SABT and SFF Level Generation (see 43D163422 sheet 15)

Two levels are generated as the result of a detected fault: SABT and SFF. Level SFF stops various portions of the Processor in anticipation of servicing the fault, and level SABT initiates the "snapshot" and "brute-forced" XED sequence for the fault.

An examination of the inputs for SABT generation shows that the presence of a particular fault will generate SABT under the following conditions:

1) A particular fault is detected or a fault group level is generated (DF or GP3, etc.)

2) Faults below Group III generally require TMCH = 0 before a fault sequence can be initiated thus indicating that the Operations Unit has completed all operations pending.

3) The PRSF flip-flop has set indicating the Interface Frame and/or Control Frame has stopped in a condition to start a meaningful fault sequence. Group II and Group I faults do not require PRSF to be set. PRSF can also be set by any one of the appending faults which does not result in a strobe from the Control Unit and Interface Frame to set PRSF (BSPLOF on sheet 14).

10.5.3.1 - contd

4) The PIN flip-flop in the Control Unit is reset indicating a memory cycle has just been completed or has not yet started.

5) The Processor is not halted.

6) None of the STOP ON FAULT switches is activated.

When SABT is generated, a "snapshot" sequence for the fault is initiated. In addition to generating SABT to initiate the "snapshot" sequence, a level designated SFF is generated to stop certain areas of the Control Unit and Interface Frame in the proper state for the "snapshot" sequence. Level SFF can be generated without SABT for those cases where any one of the four STOP ON FAULT switches on the Maintenance Panel are set. This results in the Processor stopped just prior to entering the "snapshot" sequence.

10.5.3.2 Snapshot Counter (see 43D163422 sheet 18)

Generation of the "snapshot" sequence begins with SABT generating a pulse to set the PSN flip-flop. The same pulse is also started down a delay line of approximately 324 nanoseconds. As a result of SABT, the PSN, SN0, SN1 and WF flip-flops are set at intervals of approximately 324 nanoseconds. When SN0 resets, a $SN0 is generated for the first "snapshot" pulse. When SN1 resets, an $SN1 is generated which performs the second half of the "snapshot" sequence. Setting of WF results in the majority of all Processor circuits (except the Operation Unit which is initialized by $ABT) being initialized. When WF resets, $ABT is generated which regenerates a $C in the Control Frame, starting a prepare cycle for the "brute forced" XED instruction for the fault.

10.5.3.3 Fault Address Encode (see 43D163422 sheet 13)

The fault address for the particular fault is developed from a logic network for each of the five bits of the address; FA_A being the most significant bit and FA_E

10.5.3.3 - contd

being the least significant. A trailing zero is assummed in the fault address. The various levels FA_A - FA_E are strobed into a fault code register concurrent with a \$C·WF which occurs at the end of the "snapshot" sequence. The contents of the fault code register are saved to be stored by an SCU instruction.

11.0 ASSOCIATIVE MEMORY UNIT DETAILED OPERATION

In order to implement paging, relocation, and protection of user programs the GE-645 operating system generates and maintains Descriptor Segment and Page Tables. Each time a user program attempts to access memory these tables must be referenced to obtain the absolute core location and to determine the users access rights. This procedure requires two or three memory accesses prior to an instruction or an operand fetch. In order to reduce this overhead, an Associative Memory consisting of 16 hardware registers is provided. It accepts as a reference either the effective base register (pointer), or the pointer and page number, and produces the page table location, or the block location of the page respectively. Each register has a Usage Counter (4 bits) that indicates the recency of its use with respect to the other registers. To supplement the above actions the GE-645 instruction repertoire provides the operating system three instructions for controlling the Associative Memory.

11.1 FUNCTION

The Associative Memory is constructed around sixteen 60-bit Associative Registers (ARn). The breakdown of each register is as follows:

Bits 0-17	Address Field
Bit 18	Base Bit
Bits 19-26	Boundary/Page No. Field
Bits 27-35	Descriptor Field
Bits 36-53	Pointer Field
Bit 54	Empty/Full
Bit 55	Adjust Usage Count
Bits 56-59	Usage Counter

An Associative Register may contain either a Segment Descriptor Word (SDW) or a Page Table Word (PTW). The meanings of the above fields may vary depending upon which type word is stored in that register.

11.1.1 SDW

Address Field -- It is always modulo 64. If the segment is not paged the address is the base address of the segment. If the segment is paged the address is the base address of the Page Table.

Base Bit -- It must be a "1" (a "1" means that this is a SDW)

Boundary/Page No. Field -- This is the segment boundary. It contains the number of pages in the segment.

Descriptor Field -- Each bit has its own meaning.

27 if a "1" the segment consists of 64 word pages, and if a "0" it consists of 1024 word pages.

11.1 - contd

28 if a "1" the segment is not paged, and if a "0" it is paged.

29 not used.

30 if a "1" the information stored in that segment may be altered (that is, written into), and if a "0" it may not be altered by a slave procedure.

31 if a "1" this segment may be accessed (that is, read from) in Slave mode, and if a "0" it may be accessed only in Master mode.

32 not used.

33-35 contain a code that describe the type of segment as follows:

001 - Data--This segment contains only data and may not be accessed to obtain instructions.

011 - Procedure Only--This segment is to be only self accessed or accessed by Master Procedure.

010 - Procedure Slave--This segment may contain both data and instruction. There are no restrictions on access, other than the access and write permits.

100 - Procedure Master--This segment is the same as Procedure Slave except that access is restricted to Master mode, and privileged instructions may be specified.

11.1.1 - contd

Pointer Field -- This is the SDW identification. It contains the location (page number and word number of the Descriptor Segment) relative to the Descriptor Segment Base Address from which the SDW was originally obtained.

Empty/Full -- If this bit is a "1" it indicates that the information in this register is valid, and if it is a "0" it indicates that the register has been cleared and the information in it is no longer valid.

Adjust Usage Counter -- This bit is used for internal control only. Whenever it is a "1" the Usage Counter of that register must be decremented by "1." When stored by SAM or SAZ it should appear as a "0."

Usage Counter -- This counter is used to keep track of the order in which the registers have been used. The last used register will have a count of 15, the next last used register will be 14, and on down to zero.

11.1.2 PTW

Address Field -- It is always modulo 64. If the page is a small page all 18 bits will be the base address of the page. If it is a large page bits 0-13 will be the base address of the page, which in effect makes the address modulo 1024.

Base Bit -- It must be a "0."

Boundary/Page No. Field -- This contains the page number of the page referenced by this PTW. It is part of the identification of this PTW.

11.1.2 - contd

Descriptor Field -- All bits in this field have the same meaning as for a SDW except bit 29. If bit 29 is a "1" information in this page has been altered (the page has been written into).

Pointer Field -- This, along with the Page No. Field is the identification of this PTW. It contains the location of the SDW that pointed to this PTW.

All other fields are the same for a SDW.

If the Associative Register contains a SDW (bit 18 = "1"), then bits 0-17 and 19-35 hold the SDW information and bits 36-53 hold the associative address. If the AR contains a PTW (bit 18 = "0"), then bits 0-17 and 27-35 hold the PTW information and bits 19-26 and 36-53 hold the associative address.

11.1.3 ASSOCIATIVE MEMORY CYCLES

In order to carry out its tasks, the Associative Memory must execute the following cycles:

Search
Write
Read
Adjust Usage Counters
Clear

Search -- Any time the Control Frame attempts to obtain an instruction, an operand, or an indirect word from core memory, except in Absolute mode with bit 29 off, the Associative Memory will be interrogated to see if it contains the required SDW or PTW. The search cycle first determines if any of the 16 ARs contain a pointer supplied by the Base Frame. If it finds a match it determines if the match is with a SDW (bit 18 a 1) or with PTW (bit 18 a 0). If it is a PTW, it will compare the PTW page number with the page number supplied by the

11.1.3 - contd

Control Frame. If they are identical, the Associative Memory sends the PTW to the Interface Frame and tells it to use this information to fetch the instruction, operand, or indirect word. If the page numbers are not identical, it reverts to a SDW search. If the match is a SDW, it checks to see if the segment is paged. If the segment is not paged, the Associative Memory sends the SDW to the Interface Frame and tells it to use this information to fetch the instruction, operand, or indirect word. If the segment is paged, the Associative Memory sends the SDW to the Interface Frame and tells it to use this information to fetch the proper PTW. If a match is found, bit 54 is checked to see if the information is valid. If it is not valid, the match will be ignored. If a read out occurs, the information will remain on the data out lines until the Interface Frame signifies that it has received it. If the proper PTW or SDW is not found, the Associative Memory will notify the Interface Frame of this fact and terminate its cycle.

Write -- Any time the Interface Frame receives a PTW or a SDW from core memory and no Directed Fault is detected that word will be written into the Associative Memory along with the proper identification. The Associative Memory will start this cycle when notified by the Interface Frame, which will also indicate what kind of a word is to be stored. The Associative Memory will then select the AR with a usage count of zero. If the incoming word will be a SDW it will be stored in that AR along with the pointer from the Base Frame. If the incoming word will be a PTW it will be stored in that AR along with the pointer from the Base Frame and the page number from the Control Frame. In both cases the proper control bits will be set or reset. When the PTW or SDW is available the Interface Frame will notify the Associative Memory of this fact via a strobe. This strobe initiates the actual write operation. The Write Cycle is always followed by an Adjust Usage Counter Cycle.

11.1.3 - contd

Read -- A Read Cycle will be entered if a match is found during a Search Cycle, or if a Store Associative Memory (SAM) or Store Associative Memory Zero (SAZ) command is executed. If a match is found during a Search Cycle the Associative Memory will initiate the Read Cycle itself. It will place the contents of bits 0-35 of the selected AR on the data lines to the Interface Frame, and at the same time it will generate a Strobe Start Fetch. During a SAM or SAZ instruction the Base Frame initiates the Read Cycle by placing a "1" upon the proper command line, and at the same time it supplies the information as to which AR should be read. All 60 bits of the selected AR will be placed on the data out lines to the Interface Frame.

Adjust Usage Counters -- This cycle will be entered any time an AR is read out due to a match, or following any Write Cycle. During this cycle any AR that has a usage counter higher than the count of the AR that contains the matched word or the new word will have its count decremented by one. The active AR will have its count maximized (15).

Clear -- This cycle will be initiated by a CAM or LDBR instruction causing the Base Frame to place a "1" on the Clear command line. All 16 Empty/Full bits will be reset to "0," and all Usage Counters will be set to their initial values.

11.1.4 ASSOCIATIVE MEMORY INSTRUCTIONS

The GE-645 can execute three instruction that directly affect the Associative Memory. They are:

Store Associative Memory Zero (SAZ)
Store Associative Memory (SAM)
Clear Associative Memory (CAM)

All three instructions can only be executed in Master mode.

11.1.4 - contd

Store Associative Memory Zero (SAZ) -- This instruction will cause the AR with a usage count of zero to be stored in the core location specified by the instruction (Y) and that location plus on (Y+1) as follows:

Y	0 - 35	AR 0 - 35
Y + 1	0 - 23	AR 36 - 59
Y + 1	24 - 35	Zeros

If the address requires appending the Associative Memory will be locked out so that its contents will not be altered before they are stored.

Store Associative Memory (SAM) -- This instruction will cause all 16 ARs to be stored in core memory. They will be stored by physical location, as determined by the initial usage count. AR0 will be stored in Y and Y+L, AR1 will be stored in Y+2 and Y+3,..., AR15 will be stored in Y+30 and Y+31. The bit placement will be the same as for the SAZ instruction. If the address requires appending the Associative Memory will be locked out so that its contents will not be altered before they are stored.

Clear Associative Memory (CAM) -- This instruction will clear out the Associative Memory by resetting all Empty/Full bits to 0 and setting all Usage Counters to their initial value.

11.1.5 INTERFACE SIGNALS

The Associative Memory receives signals from the Base Frame, Control Frame, and Interface Frame. It sends signals to the Interface Frame. (See Fig. 11.1.)

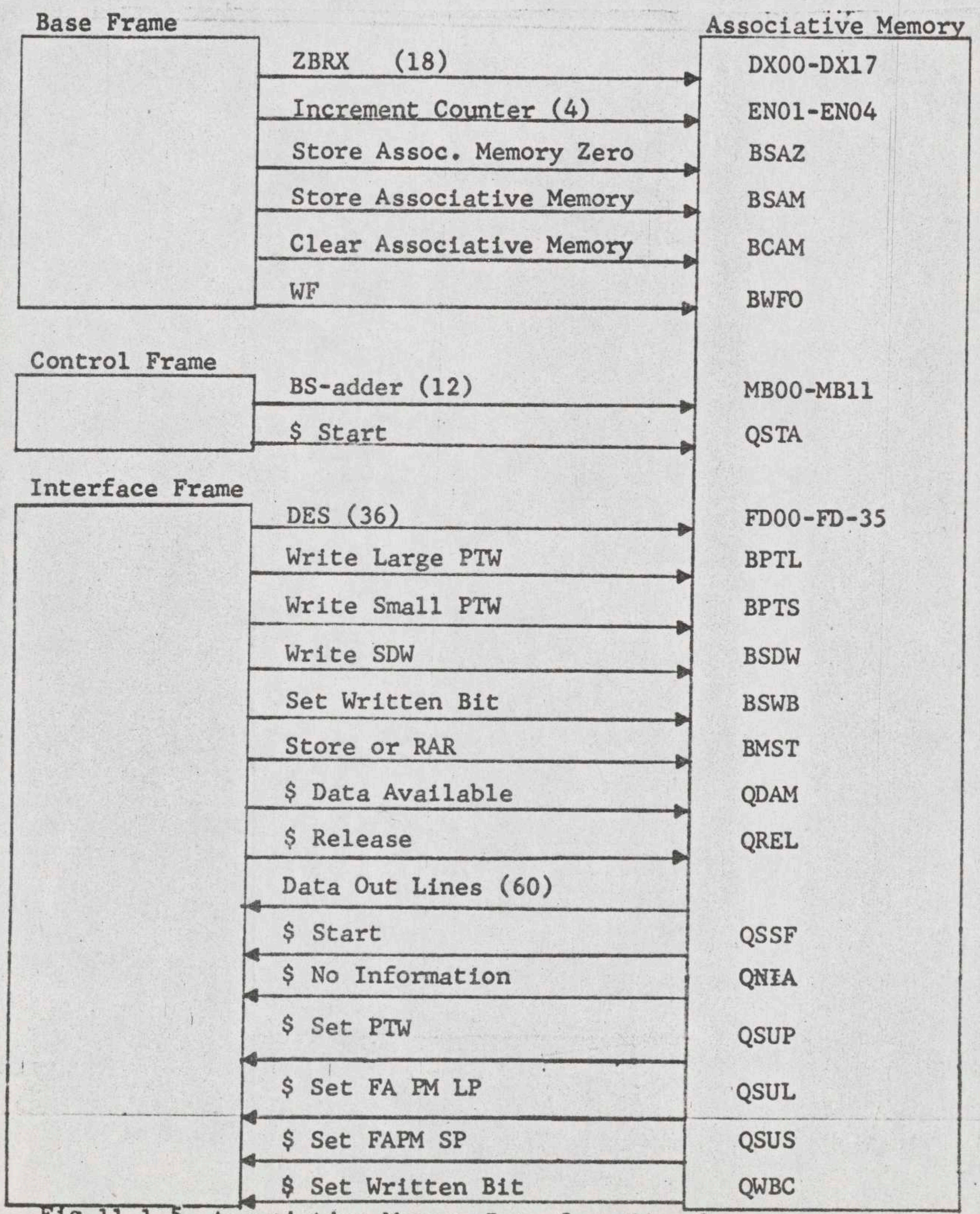

Base Frame
Associative Memory
ZBRX (18)
DX00-DX17
Increment Counter (4)
EN01-EN04
Store Assoc. Memory Zero
BSAZ
Store Associative Memory
BSAM
Clear Associative Memory
BCAM
WF
BWFO
Control Frame
BS-adder (12)
MB00-MB11
$ Start
QSTA
Interface Frame
DES (36)
FD00-FD-35
Write Large PTW
BPTL
Write Small PTW
BPTS
Write SDW
BSDW
Set Written Bit
BSWB
Store or RAR
BMST
$ Data Available
QDAM
$ Release
QREL
Data Out Lines (60)
$ Start
QSSF
$ No Information
QNIA
$ Set PTW
QSUP
$ Set FA PM LP
QSUL
$ Set FAPM SP
QSUS
$ Set Written Bit
QWBC

Fig.11.1.5 Associative Memory Interface Signals

11.1.6 MANUAL OPERATION

The operation of the Associative Memory can be modified by switches on the Maintenance Panel. If the Test Enable Signal is true and the TEST CLOCK switch is in the FAST position the three delay lines that produce QP111, QP221, and QP331 will be shortened. If the Test Enable signal is true and the SF ENABLE switch is in the STEP position the operation of the search cycle is interrupted. It will stop just before the generation of QP111, or QP221, or QP331. Depressing the SF STEP switch will generate the required pulse and the search cycle will continue up to, but not including the next pulse. To return to normal operation the SF ENABLE switch must be returned to normal and the SF STEP depressed once.

11.2 THEORY OF OPERATION

In order to understand the operation of the Associative Memory, an understanding of the philosophy of segmentation, paging, and how it is implemented in the GE-645 is required. This discussion assumes that the reader has this knowledge. The operation of the Associative Memory can be divided into four parts. These parts will be taken up separately in this order: Interface, Basic Operating Cycles, Instructions, and Manual Operation.

11.2.1 INTERFACE

The Interface between all affected frames in the Processor and the Associative Memory are listed by EDA mnemonic in Tables 11.2.1 and 11.2.1a.

Table 11.2.1 Associative Memory Inputs

Mnemonic	Description
	Base Frame
BCAMF (CAM)	CLEAR ASSOC.MEMORY
BSAMF (SAM)	STORE ASSOC.MEMORY
BSAZF (SAZ)	STORE ASSOC.ZERO
DX00F-DX17F $(\overline{\text{ZBRX0-17}})$	BASE REGISTER EXTERNAL BUS
EN01F-EN04F (INC1-4)	INCREMENTOR REGISTER
EWFOF (WF)	WF ABORT F/F
	Control Frame
MB00B-MB11B (BS0-11)	ADDRESS BASE ADDER
QSTAB ($	START ASSOC.MEMORY PULSE
	Interface Frame
BMSTM	DECODE FOR STORE OR RAR CYCLES TO ASSOC.MEMORY
BPTLM (WPTWL)	WRITE PTW LARGE
BPTSM (WPTWS)	WRITE PTW SMALL
BSDWM (WSDW)	WRITE SDW
ESWBM	
FD00M-FD35M (DES0-35)	36-BIT DESCRIPTOR REGISTER
QDAMM ($DAAM)	DATA AVAILABLE PULSE
QRELM ($RELEASE)	RELEASE PULSE
	Maintenance Panel
SQS1T	TEST CLOCK SWITCH
SQS1T	TEST CLOCK SWITCH
SQS2T	TEST CLOCK SWITCH
SQS2T	TEST CLOCK SWITCH
SSSET	ENABLE STEP SWITCH
SSSET	ENABLE STEP SWITCH
SSSTT	STEP SWITCH
SSSTT	STEP SWITCH
STSTT	TEST SWITCH
STSTT	TEST SWITCH

Table 11.2.1a Associative Memory Outputs

Mnemonic	Description
Interface Frame	
DC30E-DC35E (ZADS30-35)	ASSOC. REG. DESCRIPTOR BUS
BA55E (ZAU55)	ASSOC. REG. ADJUST USAGE BIT BUS
BE54E (ZEF54)	ASSOC. REG. EMPTY/FULL BIT BUS
BU56E-BU59E (ZSUC56-59)	SELCTD. USAGE COUNTER BUS
DA00E-DA17E (ZASM0-17)	ADDRESS MOD. 64 BUS
DB18E (ZBB18)	BASE BIT BUS
DB19E-DB26F (ZABD19-26)	BOUNDARY BUS
DC27E-DC29E (ZADS27-29)	ASSOC. REG. DESCRIPTOR BUS
DT36E-DT53E (ZPTR50-53)	ASSOC. POINTER BUS
QNIAE ($NIA)	NO INFORMATION AVAILABLE PULSE
QSSFE ($SF)	START FETCH PULSE
QSULE ($SET FA PMLP)	SET UP FAPM (LP) FETCH PULSE
QSUPE ($SET PTW)	SET UP PTW FETCH PULSE
QSUSE ($SET FA PMSP)	SET UP FAPM (SP) FETCH PULSE
QWBCE ($START WBC)	START WRITTEN BIT CYCLE
Maintenance Panel	
EAUVE	ADJUST USAGE VALUE F/F
EPSHE	POINTER SEARCH F/F
EPSLE	PTW (LARGE) SEARCH F/F
EPSSE	PTW (SMALL) SEARCH F/F
ERELE	RELEASE F/F
ESCRE	STROBE COUNTER RELEASE F/F
ESDPE	SDW PRESENT F/F
ESDSE	SDW SEARCH F/F
Relay Box	
BA55E (ZAU55)	ASSOC. REG. ADJUST USAGE BIT BUS
BE54E (ZEF54)	ASSOC. REG. EMPTY/FULL BIT BUS
BU56E-BU59E (ZSUC56-59)	SELCTD. USAGE COUNTER BUS
DA00E-DA17E (ZASM0-17)	ADDRESS MOD. 64 BUS
DB18E (ZBB18)	BASE BIT BUS
DB19E-DB26E (ZABD19-26)	BOUNDARY BUS
DC27E-DC35E (ZADS27-35)	ASSOC. REG. DESCRIPTOR BUS
DT36E-DT53E (ZPTR50-53)	ASSOC. POINTER BUS

11.2.2 BASIC OPERATING

There are five basic operating cycles and a terminate condition.

1) Search Cycle initiated by the Control Frame

2) Write Cycle initiated by the Interface Frame

3) Read Cycle initiated by the Associative Memory or Base Frame

4) Adjust Usage Counters Cycle initiated by the Associative Memory

5) Clear Cycle initiated by the Base Frame

6) Termination

11.2.2.1 Search Cycle

Any time the Processor performs an appending operation on an address, the Associative Memory is searched to see if it contains the information necessary for this operation. There are ten possible combinations of matches or no matches. These are shown in Fig. 11.2.2.1 along with the pulses that will be generated and sent to the Interface Frame for each combination. Following the test on the Search Cycle are several sample timing diagrams.

The Control Frame prepares the address to be appended. When it determines that the address will be ready by the time it is needed it issues a QSTA1 to the Associative Memory and a \$ Append to the Interface Frame. At this time the Base Frame will have placed the pointer on the DX00-DX17 (ZBRX$_{0-17}$) lines.

Table 11.2.2.1 Search Possibilities vs. Pulses Generated and Sent to Interface Frame

Pointer Match	Non-paged SDW	Paged SDW Large	Paged SDW Small	PTW Large	PTW Small	QNIA1	QSUL1	QSUS1	QSUP1	QSSF1
No						X				
Yes Large Page						X	X			
Yes Small Page						X		X		
Yes	Yes									X
Yes		Yes					X		X	X
Yes			Yes					X	X	X
Yes		Yes		Yes			X			X
Yes			Yes		Yes			X		X
Yes				Yes			X			X
Yes					Yes			X		X

11.2.2.1 - contd

QSTA1 starts the Search Cycle by entering a delay line, from which it will emerge as QP111 in 216 nanoseconds. During this interval of time the pointer search is accomplished. The information on the DX00-DX17 lines is compared with each Associative Register (A through S), bits 36 through 53. If any ARs pointer matches DX00-DX17 and its Empty/Full bit is a "1" a level will come true that will switch that AR onto the data-out lines and will cause BGMA1 to become true. In the case where more than one AR has a match the output of the matching ARs will be ORed together on the data out lines.

QP111 when it emerges from the delay line will do the following:

1) Reset EPSH if there is a match with a paged SDW or a PTW.

2) Set ESDP if there is a match with a SDW.

3) Send a QSSF1 pulse to the Interface Frame if there is a match with a nonpaged SDW.

4) Send a WSUL1 pulse to the Interface Frame and set EPSL if there is a match with a large paged SDW or a large PTW.

5) Send a QSUS1 pulse to the Interface Frame and set EPSS if there is a match with a small paged SDW or a small PTW.

6) Send a QNIA1 pulse to the Interface Frame if there is no match.

11.2.2.1 - contd

QP111 will enter another delay line if there is a match with a paged SDW or a PTW. It will emerge in 234 nanoseconds as QP211. If a QNIA pulse was generated the Associative Memory will wait until the Interface Frame has obtained the required SDW and upon signal it will enter a Write Cycle. If a QSSF1 pulse was generated a Read Cycle will be entered.

During the interval of time between QP111 and QP211 the PTW search is performed. The exact nature of this search will depend upon whether it is looking for a large or small PTW.

EPSL set (search for large PTW)

In this case DX00-DX17 is compared against bits 36-53 of all ARs and MB00-MB07 is compared against bits 19-26 of all ARs. If a match exists, the Empty/Full bit is a "1," and the Base bit is a "0" a level will come true that will switch that AR onto the data out lines and BGMA1 will come true.

QP211 when it emerges from the delay line will do the following:

1) Send a QSSF1 pulse to the Interface Frame if a match is found and either the operation is not a Write or RAR or the Written bit of the PTW is a "1."

2) Send a QSUP1 pulse to the Interface Frame, set ESDS, reset EPSL, and reset EPSS if there is no match, but there is a SDW present.

3) Send a QWBC1 pulse to the Interface Frame if there is a match, the operation is a write or RAR, and the Written bit is a "0."

11.2.2.1 - contd

4) Send a QNIA1 pulse to the Interface Frame if there is no match and there is no SDW present.

QP211 will enter another delay line if there is no match, but there is a SDW present. It will emerge in 216 nanoseconds as QP311.

EPSS set (search for small PTW)

In this case DX00-DX17 is compared against bits 36-53 of all ARs and MB04-MB11 is compared against bits 19-26 of all ARs. If a match exists, the Empty/Full bit is a "1," and the Base bit is a "0" a level will come true that will switch that AR onto the data out lines and BGMA1 will come true.

QP211 when it emerges from the delay line will do the same things as when EPSL is set.

In either case if a QNIA1 was generated the Associative Memory will wait until the Interface Frame has obtained the required SDW and upon signal it will enter a Write Cycle. If a QSSF1 was generated a Read Cycle will be entered.

During the interval of time from QP211 to QP311 a SDW search will be performed. Because of the conditions required to enter this portion of the search cycle it is known that a match will be made. The problem is to find the AR that contains the right SDW. This is done by comparing DX00-DX17 against bits 36-53 of all ARs. A match will exist with one AR that also has its Empty/Full bit a "1" and its Base bit a "1," which will cause a level to come true switching that AR onto the data out lines, and BGMA1 will come true.

11.2.2.1 - contd

QP311 when it emerges from the delay line it will send a QSSF1 to the Interface Frame and the Read Cycle will be entered. This completes the Search Cycle.

There is a special case dealing with the Written bit. Any time a page in core memory is written into by a store Instruction or a RAR operation this fact is remembered by setting the Written bit to a "1" in both the PTW in core memory and the PTW in Associative Memory. This operation is necessary only on the first occasion of the page being written into. If a PTW with the Written bit a "0" is obtained from core memory for a store or RAR the Interface Frame will handle the setting of the Written bit of the PTW in core memory and will set the Written bit in the PTW before it is sent to the Associative Memory to be stored. When a PTW match is found in the Associative Memory the BMST1 line from the Interface Frame and the Written bit are checked. If the BMST1 line is a "1" and the Written bit is a "0" the Associative Memory will send a QWBC1 pulse to the Interface Frame. The Interface Frame will place a "1" on the BSWB1 line and then set the Written bit of the PTW in core memory. A "1" on the BSWB1 line will cause bit 29 (Written bit) of the AR containing that PTW to be set to a "1."

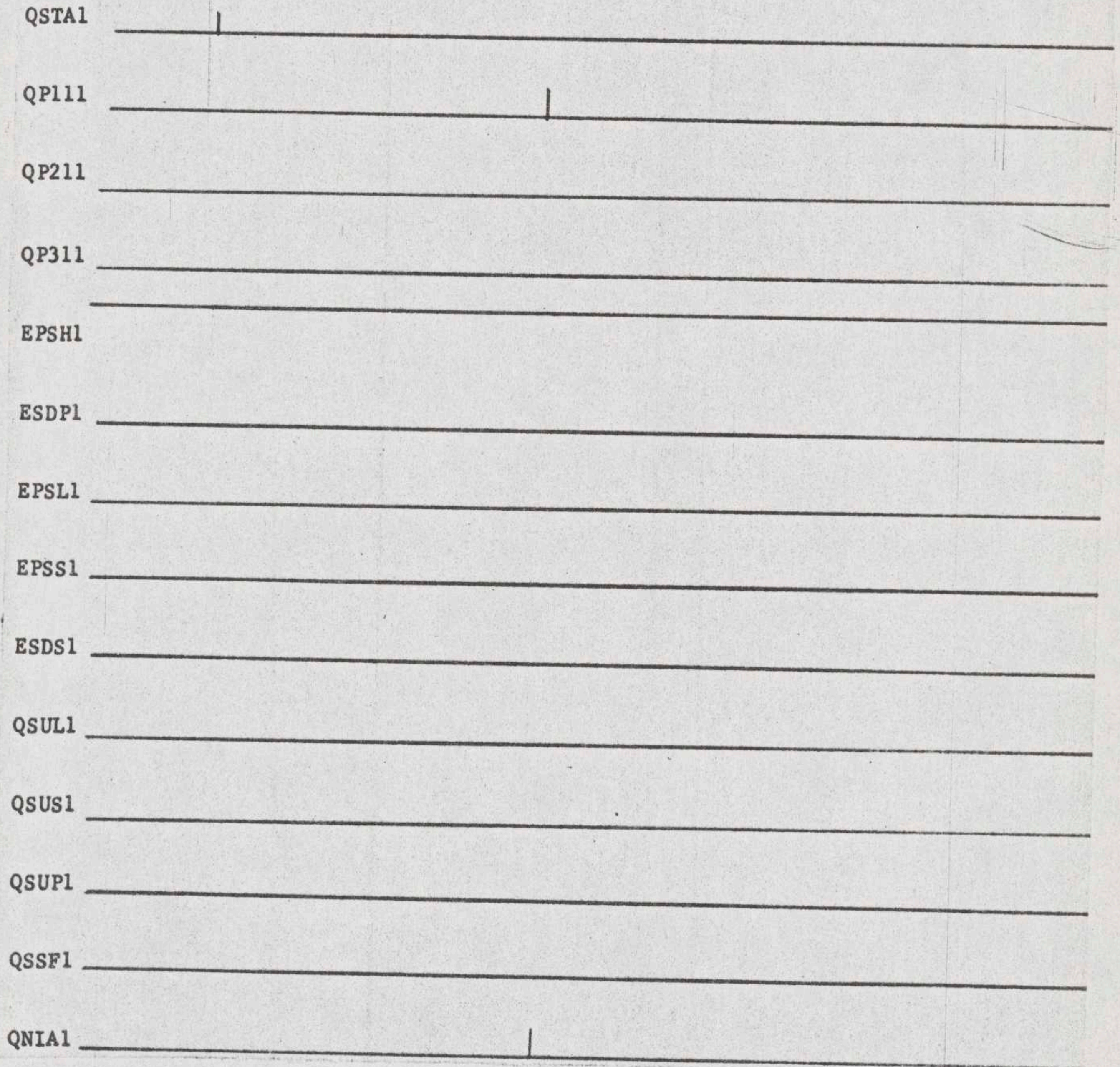

QSTA1
QP111
QP211
QP311
EPSH1
ESDP1
EPSL1
EPSS1
ESDS1
QSUL1
QSUS1
QSUP1
QSSF1
QNIA1

Fig. 11.2.2.1 No Match, Timing Diagram

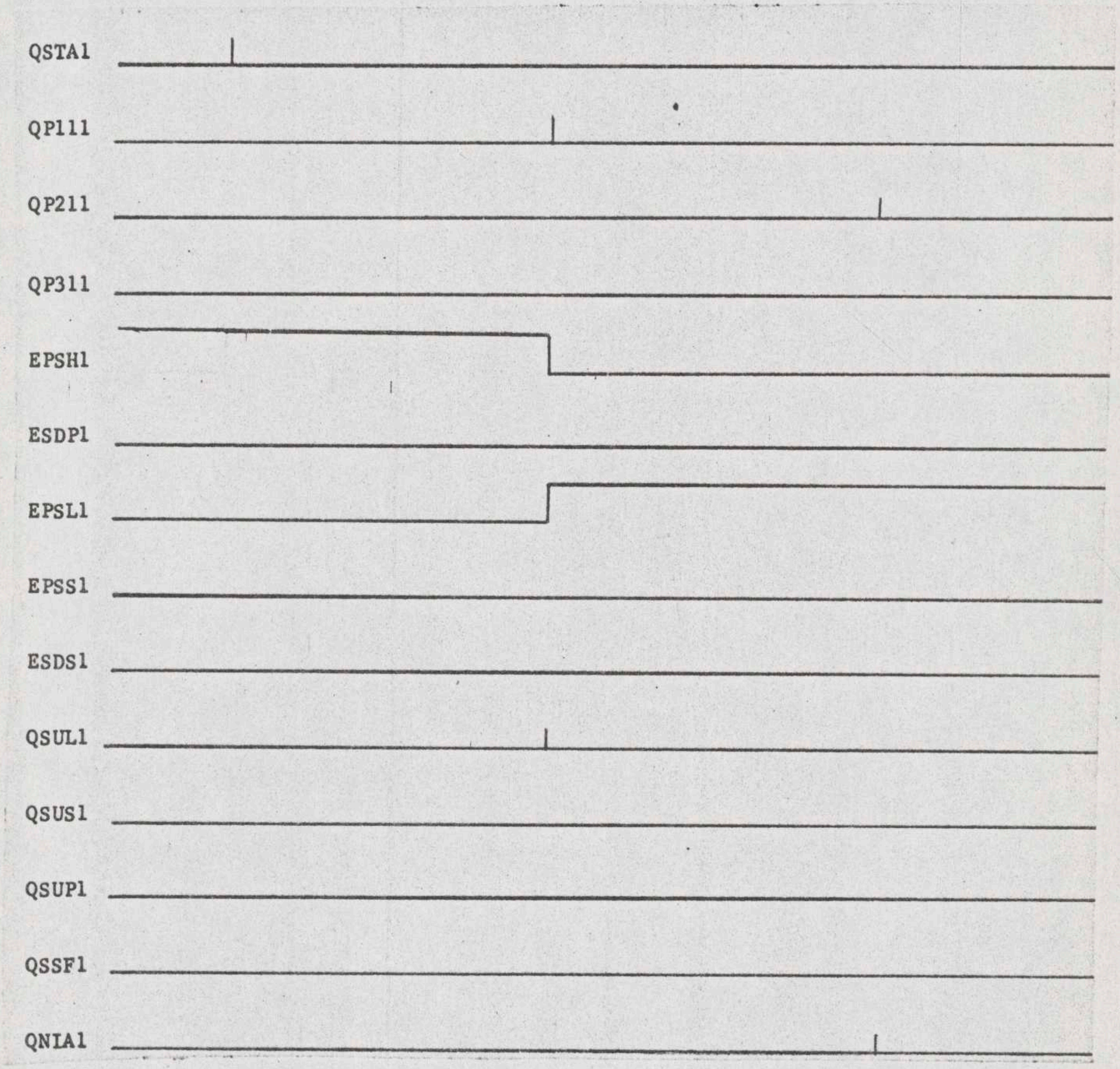

QSTA1
QP111
QP211
QP311
EPSH1
ESDP1
EPSL1
EPSS1
ESDS1
QSUL1
QSUS1
QSUP1
QSSF1
QNIA1

Fig. 11.2.2.1a Pointer Match for Large Page, No SDW or PTW Match, Timing Diagram

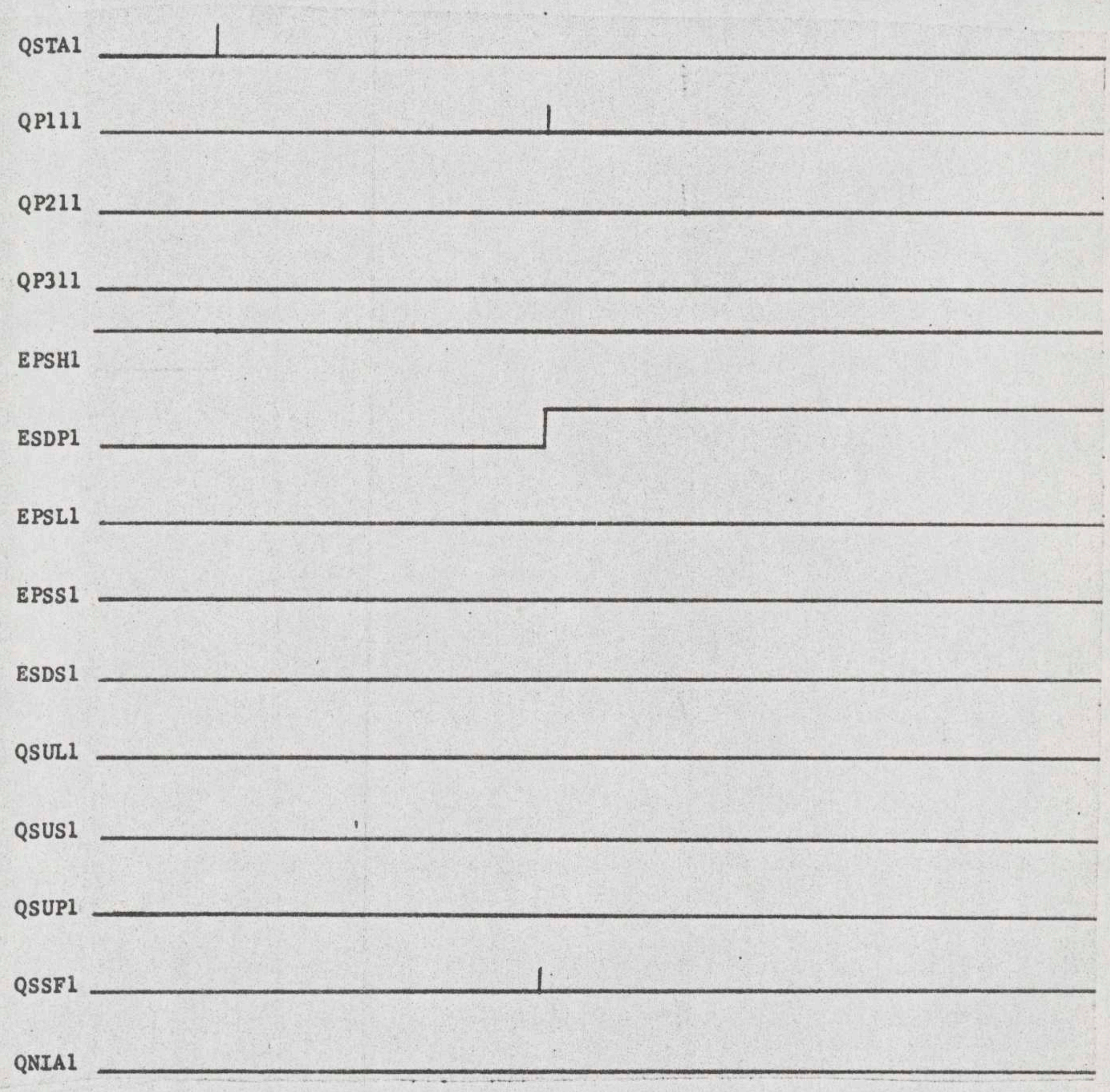

QSTA1
QP111
QP211
QP311
EPSH1
ESDP1
EPSL1
EPSS1
ESDS1
QSUL1
QSUS1
QSUP1
QSSF1
QNIA1

Fig. 11.2.2.1b SDW Match (nonpaged), Timing Diagram

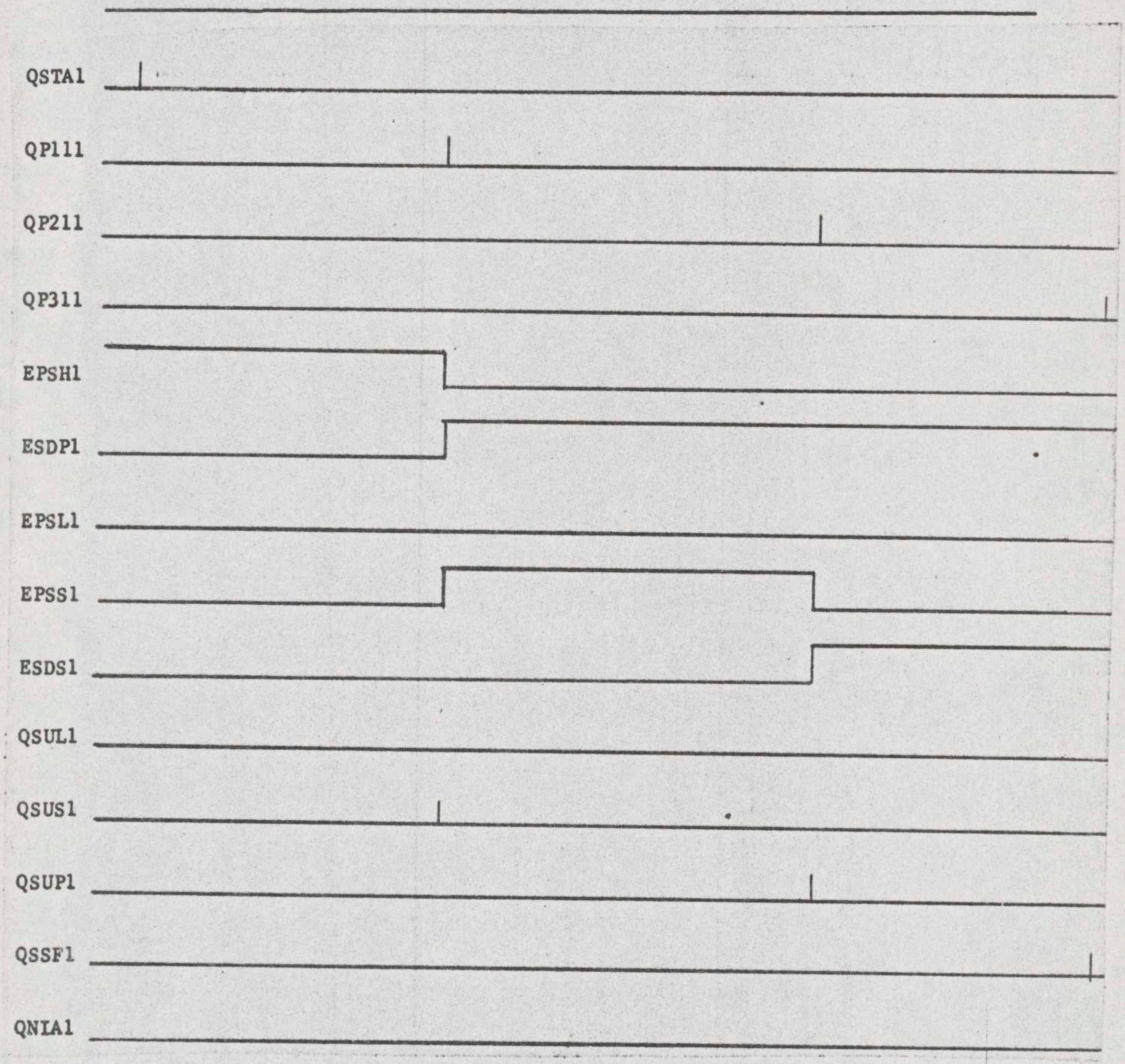
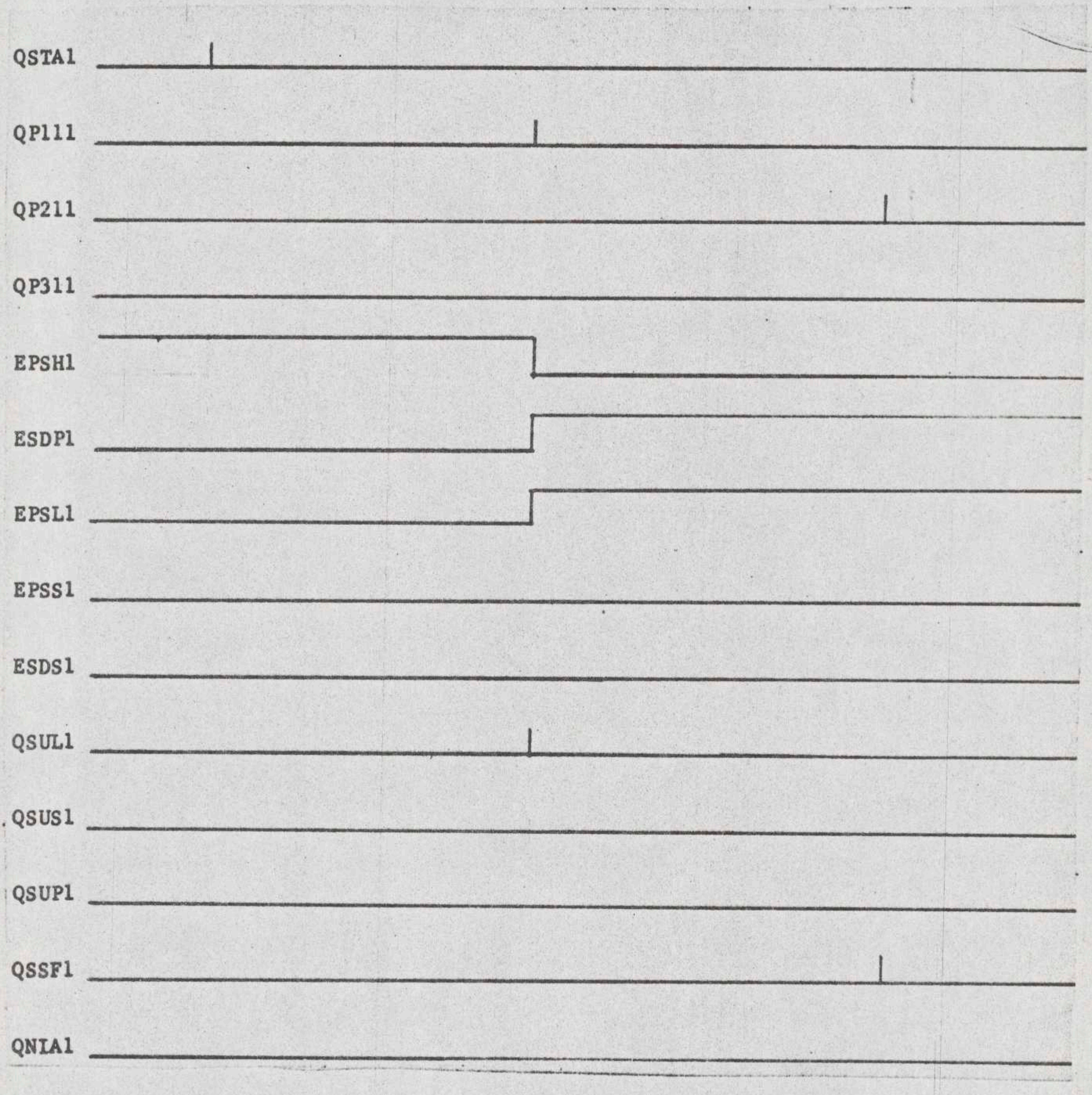
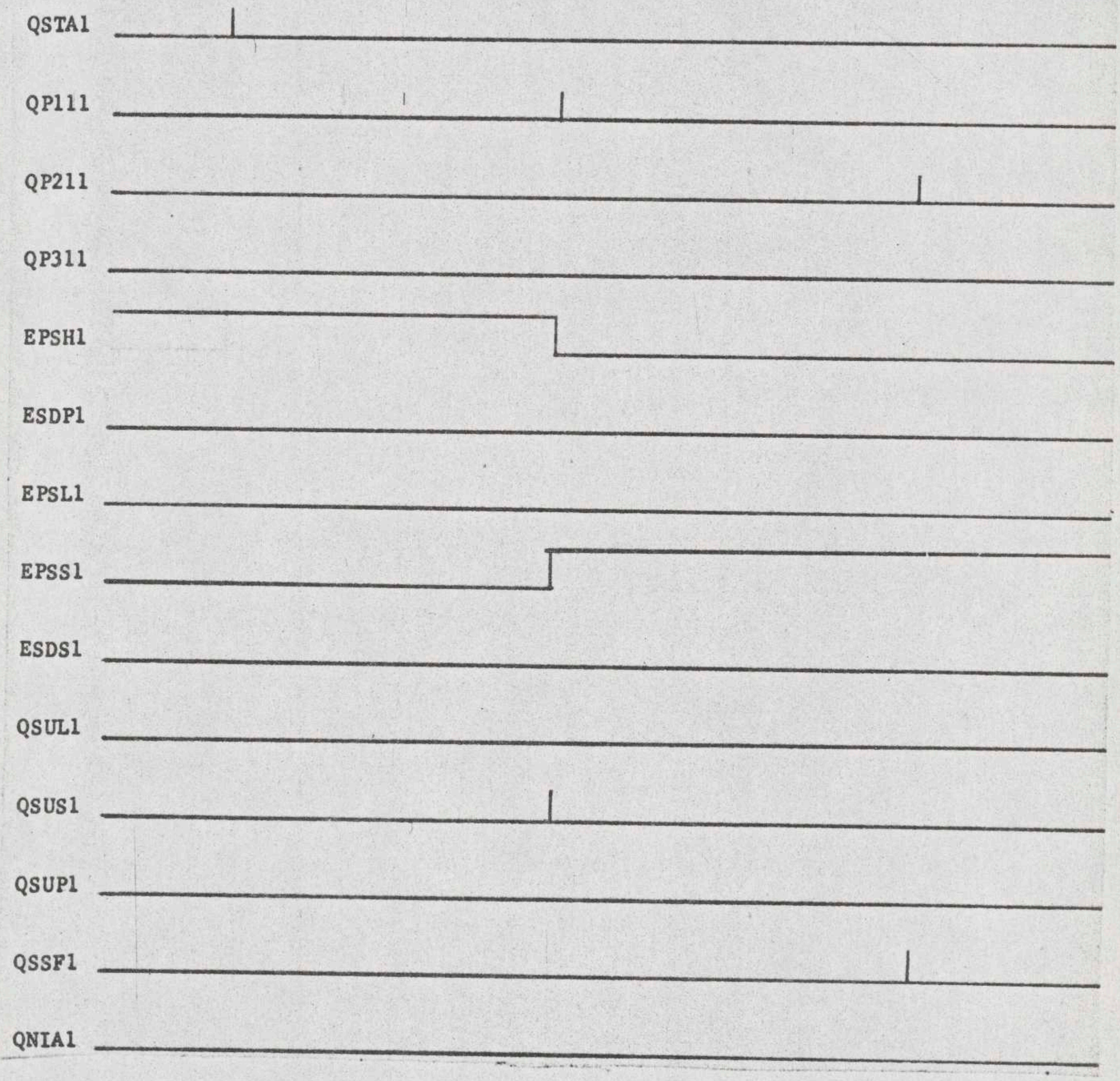

QSTA1
QP111
QP211
QP311
EPSH1
ESDP1
EPSL1
EPSS1
ESDS1
QSUL1
QSUS1
QSUP1
QSSF1
QNIA1

Fig. 11.2.2.1c Match SDW for Small Page, Timing Diagram


QSTA1
QP111
QP211
QP311
EPSH1
ESDP1
EPSL1
EPSS1
ESDS1
QSUL1
QSUS1
QSUP1
QSSF1
QNIA1

NOTE - Written bit of PTW = "1"

Fig. 11.2.2.1d Match SDW for Large Page and PTW, Timing Diagram


QSTA1
QP111
QP211
QP311
EPSH1
ESDP1
EPSL1
EPSS1
ESDS1
QSUL1
QSUS1
QSUP1
QSSF1
QNIA1

— Fig. 11.2.2.1e Match PTW for Small Page No SDW, Timing Diagram

11.2.2.2 Write Cycle

A Write Cycle will store a PTW or a SDW in the Associative Register that has a usage count of zero. The Interface Frame upon obtaining a SDW or a PTW from core memory will store it in the Descriptor Register, and will check the descriptor field for a Directed Fault. If there is no fault it will update the descriptor field and place a 1 on one of three command lines to the Associative Memory. The command line selected depends upon the type of word to be stored. The three command lines are:

1) BSDW1 - Write a SDW

2) BPTL1 - Write a large PTW

3) BPTS1 - Write a small PTW

To write into the Associative Memory the Interface Frame will place the contents of the Descriptor Register on the FD00-FD35 lines and a 1 on a command line. When a command line goes high the Associative Memory will switch the input lines to all 16 ARs as per Table 11.2.2.2. The Interface Frame will then place a pulse on the QDAM1 line, which will select the AR with a usage count of zero and issue a strobe to that AR only. This strobe will write the input information into the selected AR. The Adjust Usage Counters Cycle will be entered at this point.

11.2.2.2 Write Cycle

A Write Cycle will store a PTW or a SDW in the Associative Register that has a usage count of zero. The Interface Frame upon obtaining a SDW or a PTW from core memory will store it in the Descriptor Register, and will check the descriptor field for a Directed Fault. If there is no fault it will update the descriptor field and place a 1 on one of three command lines to the Associative Memory. The command line selected depends upon the type of word to be stored. The three command lines are:

1) BSDW1 - Write a SDW

2) BPTL1 - Write a large PTW

3) BPTS1 - Write a small PTW

To write into the Associative Memory the Interface Frame will place the contents of the Descriptor Register on the FD00-FD35 lines and a 1 on a command line. When a command line goes high the Associative Memory will switch the input lines to all 16 ARs as per Table 11.2.2.2. The Interface Frame will then place a pulse on the QDAM1 line, which will select the AR with a usage count of zero and issue a strobe to that AR only. This strobe will write the input information into the selected AR. The Adjust Usage Counters Cycle will be entered at this point.

Table 11.2.2.2 Associative Register Bit Positions vs. Input Data Lines

Associative Reg. Bits	SDW	Large PTW	Small PTW
00-17	FD00-FD17	FD00-FD17	FD00-FD17
bit 18	1	0	0
19-26	FD19-FD26	MB00-MB07	BM04-MB11
27-35	FD27-FD35	FD27-FD35	FD27-FD35
36-53	DX00-DX17	DX00-DX17	DX00-DX17

11.2.2.3 Read Cycle

The purpose of the Read Cycle is to place the information contained in the selected Associative Register on the data out lines. This can happen during a Search Cycle, during a SAZ instruction, or during a SAM instruction.

Search Cycle -- If a match occurs during any one of the three parts of a Search Cycle, a signal indicating the AR that produced the match will come true. This match signal will produce the read out signal for the corresponding AR. During a Search Cycle only bits 0-35 and 54-59 are read out.

SAZ Instruction -- The SAZ signal is ANDed with the Usage Count Equal Zero level to produce the read out signal for that AR. All 60 bits of the AR are read out.

SAM Instruction -- The SAM signal allows the Increment Counter in the Base Frame to select the AR to be read out. A count of zero will read out of AR A, a count of one will read out of AR B, etc. All 60 bits of the selected AR will be read out.

11.2.2.3 - contd

In all cases the information will remain on the data out lines as long as the conditions that produced the read out exists.

11.2.2.4 Adjust Usage Counters Cycle

Any time an Associative Register is read out of because of a match during a Search Cycle or written into, the Usage Counters must be adjusted. There is a 4-bit Usage Adder for each AR. One set of inputs to each adder is always bits 56-59 of its corresponding AR. The other set of inputs is variable. When a Read or Write Cycle occurs, the signal that reads the AR onto the data out lines will also switch the complemented outputs of bits 56-59 of the data-out lines to the variable set of inputs to all 16 four-bit adders. At this time EAUV is reset causing a carry to be added to the least significant bit of all adders. The net result is that the usage count of the selected AR is subtracted from the usage count of all ARs. Only the ARs that have a count equal to or greater than the selected AR will have a carry out of their adder.

During a Search Cycle if a match is found, the AR that produced the match will be read out onto the data out lines and a QSSF1 pulse sent to the Interface Frame. At the same time a pulse will enter a delay line to emerge 90 nanoseconds later as ASAU1. From QSSF1 to QSAU1 the adders are performing the subtraction as indicated above. QSAU1 will set EAUV and will strobe the Adjust Usage Counter bits (55) of all ARs. Any adder that produced a carry will allow QSAU1 to set the Adjust Usage Counter bit of its corresponding AR. Therefore, the selected AR and all ARs that have a higher usage count will have their Adjust Usage Counter bit set. All other ARs will have their Adjust Usage Counter bit remain reset. QSAU1 will also enter a delay line to emerge 270 nanoseconds later as QSCT1.

11.2.2.4 - contd

When EAUV set it placed all ones on the set of variable inputs to all adders and removed the carry input to the least significant bit. EAUV set will also cause the control lines for the adder of the selected AR to set that adder up for an OR operation instead of an ADD operation. The outputs of the adders will now be all ones (a count of 15) for the selected AR and the original count minus one for all other ARs. The output of each adder is routed back to its corresponding AR's Usage Counter. QSCT1 is used to strobe the Usage Counters. However, it will strobe only those Usage Counters that have their Adjust Usage Counter bit set. At the end of the Adjust Usage Counters Cycle the selected AR will have a count of 15, all ARs that had a count higher than the original count of the selected AR will have a count one less than their original count, and all other ARs will be unchanged. QSCT1 will also reset EAUV and set ESCR.

During a Write Cycle a QDAM1 pulse will be received when the data to be written is available. This pulse also enters the delay line to produce a QSAU1 pulse. The read out signal is developed for the selected AR (the one with a usage count of zero) to switch its usage count onto the data out lines. From this point on the Adjust Usage Counters Cycle proceeds as described above for the Search Cycle.

11.2.2.5 Clear Cycle

The purpose of the Clear Cycle is to initialize the Associative Register. This cycle is initiated by the Base Frame placing a "1" on the BCAM1 line. This will unconditionally reset all 16 Empty/Full bits, and it will unconditionally set the proper count in each of the Usage Counters (0000 in the Usage Counter for AR A, 0001 in the Usage Counter for AR B, 0010 in the Usage Counter for AR C, etc.). This cycle is ended when the Base Frame removes the "1" from the BCAM1 line.

11.2.2.6 Termination

The Associative Memory is started by a QSTA1 pulse. This pulse will also reset EREL and ESCR in the terminate logic. When a QSCT1 pulse is generated at the end of an Adjust Useage Counters Cycle ESCR will set. When the Interface Frame has finished with the Associative Memory it issues a QREL1 pulse, which will set EREL. These two events are asynchronous and either can occur first. Using two flip-flops ensures that both units have completed their Associative Memory operation before the Associative Memory logic is initialized. ESCR set will unconditionally reset all Adjust Usage Counter bits. When both flip-flops are set they will produce a reset signal, which will reset EAUV, ESDP, EPSL, EPSS, ESDS, and set EPSH. This initializes the control logic and the Associative Memory is now ready to start another cycle.

In the case where a paged SDW is found but not the PTW the Associative Memory will go through an Adjust Usage Counters Cycle when the SDW is read out and again when the PTW is written into an AR. The termination must wait until completion of the second Adjust Usage Counters Cycle. The first QSCT1 will set ESCR clearing the Adjust Usage Counter bits, but the QDAM1 pulse for the PTW Write Cycle will reset it. The second QSCT1 pulse will again set ESCR, and now the Associative Memory is ready to terminate.

After turn on, and at other times, it may be desirable to initialize the Associative Memory. During turn on, if the stop switch is activated, or during any fault the Control Frame will go through an Abort Cycle. During this cycle

11.2.2.6-contd

a WF signal will be produced. WF will unconditionally reset EDIS and unconditionally set EREL and ESCR, which will initialize the Associative Memory control logic.

11.2.3 ASSOCIATIVE MEMORY INSTRUCTIONS

There are three instructions that deal directly with the Associative Memory: SAZ, SAM, and CAM.

11.2.3.1 SAZ (157)

When this op code is decoded the Base Frame will place a "1" on the BSAZ1 line. This level checks the usage count of all the ARs. A usage count of zero and BSAZ1 will produce two read out levels for the corresponding AR. One read-out level will switch bits 0-35 and 54-59 onto the data out lines. The other read-out level will switch bits 36-53 onto the data-out lines until the Base Frame removes the "1" from the BSAZ1 line. The Adjust Usage Counter Cycle will not be entered. This completes the SAZ instruction.

11.2.3.2 SAM (557)

When this op code is decoded the Base Frame will place a "1" on the BSAM1 line. At the same time it will send the count in its Increment Counter minus the least significant bit. This counter will be initially set to zero and will be stepped each time a word is stored in core memory. Since each AR requires two memory words the data on the data out lines will be changed on every other memory cycle. The affects of BSAM1 and the Increment Counter on the data out lines is shown in Table 11.2.3.2. As in the SAZ instruction, two read-out levels are produced for each AR to switch all 60 bits onto the data out lines. When the last AR has been stored in core memory the Base Frame will remove the "1" from the BSAM1 line ending the SAM instruction. The Adjust Usage Counters Cycle will not be entered.

Table 11.2.3.2 Increment Counts vs. Register Stored for the SAM Instruction

Increment Counter					DATA OUT LINES	
To Assoc. Mem.				Not to A.M.		
EN04	EN03	EN02	EN01	EN00		
0	0	0	0	0	Associative Register	A
0	0	0	0	1	Associative Register	A
0	0	0	1	0	Associative Register	B
0	0	0	1	1	Associative Register	B
0	0	1	0	0	Associative Register	C
0	0	1	0	1	Associative Register	C
0	0	1	1	0	Associative Register	D
⋮	⋮	⋮	⋮	⋮	⋮ ⋮	⋮
1	1	1	1	1	Associative Register	S

NOTE

EN00 is the 2 degree position of the Increment Counter in the Base Frame and is not sent to the AM.

11.2.3.3 CAM (532)

When this op code is decoded the Base Frame will place a "1" on the BCAM1 line. This will cause the Associative Memory to perform the Clear Cycle as described under Basic Operating Cycles.

11.2.4 MANUAL OPERATION

Test Check -- The delay lines for QP111, QP221, and QP331 are controlled by a level BCDYO. When BCDYO is at + 6 volts there will be a normal delay. If the TEST ENABLE switches are in TEST and the SF TEST CLOCK switch is in the FAST position BCDYO will go to ground and all three delay lines will be shortened.

Step -- If the TEST ENABLE switches are in TEST and the SF ENABLE switch is in the ENABLE position the operation of the Search Cycle will be interrupted. The QSTA1 pulse will set EPSH and enter the delay line for QP111 as in normal operation. However, it will also set EDIS, which places an inhibit on the output of the three delay lines. The operation of the Search Cycle will proceed normally, but QP111 will not be generated. This cycle will remain in this state until the SF STEP switch is depressed. Depressing the SF STEP switch will generate a QPTM1 pulse, which samples the various states in which the Search Cycle can hang up. In this case EPSH and DEIS set will allow QPTM1 to generate a QP111 which will perform its normal functions. However, if a QP221 is required, the Search Cycle will hang up with either EPSL or EPSS set. Depressing the SF STEP switch will generate a QPTM1. EDIS set and EPSL or EPSS set will allow QPTM1 to generate a QP221, which will perform its normal functions.

11.2.4 - contd

However, if a QP331 is required the Search Cycle will hang up with ESDS set. Depressing the SF STEP switch will generate a QPTM1. EDIS and ESDS set will allow QPTM1 to generate a QP331, which will perform its normal functions. If a Search Cycle is ended by a QSSF1 or QNIA1 pulse the Processor operation will continue normally until another Search Cycle is entered. To resume normal operation of the Associative Memory the SF ENABLE switch must be returned to NORMAL, and the SF STEP switch depressed. This will produce a QPTM1 pulse, which will generate the required QP pulse and reset EDIS removing the inhibit from the delay lines.

12.1.1 - contd

d) The "N" and "H" registers which provide temporary operand storage during arithmetic operations;

e) The S adder where operand addition is performed;

f) The "A" and "Q" registers which may function separately as accumulator and quotient registers or in combination for double precision operations;

g) The G counter which counts the iterations of iterative type operations such as multiplication;

h) The exponent registers (ER & DR) and exponent adder (ES) used to store and add exponents in floating point operations;

i) Index registers which are involved in various load, store, arithmetic and logical operations find their primary use in storing information for address modification, which is handled by the control unit;

j) The indicator register, which is a collection of flip-flops indicating operation results, is also used for decision flags.

k) The timer is a 24-bit decrementing counter which, when it has decremented to zero, indicates that the operation in process has exceeded the allowed time. A timer run out flag is set when the timer decrements to zero.

12.2 OU TIMING

There are 11 basic OU timing intervals or "G" cycles available for performing all of the OU functions. These minor cycles, when sequenced in a manner designated by a machine instruction, constitute a total timing sequence which permits the OU to execute its portion of an instruction.

The 11 "G" cycles are:

GS - Setup or start cycle

GO - General operate cycle

GC - Single precision store, or the first half of a double precision store cycle.

GT - Double precision store operation cycle

GFT - Last half of a double precision store cycle

GE - Floating point exponent comparison cycle

GA - Mantissa alignment cycle

GD1 - First divide operation cycle

GD2 - Second divide operation cycle

GN - Mantissa Normalization and/or Overshift Correction cycle

GF - Final cycle

12.2-contd

The duration of each of these 11 cycles is normally one of the three basic OU times (that is, fast, medium or slow). In special cases a cycle may have a variable wait time (T_M) terminating when a data store operation has been completed.

See paragraph 12.2.2 for a description of the OU Timing Pulse Generator.

Normally nonvariable duration cycles are:

GF	Always of a slow time duration when not associated with a store instruction.
GN, GA	Always of a fast time duration.
GE, GD1, GD2	Always of a slow time duration.
GFT, GT	Always a T_M time duration.

All "G" Cycles are mutually exclusive except:

GS and GF which may overlap.

GC which occurs simultaneously with GS and GT for double precision stores, or simultaneously with GS and GF for single precision stores, or simultaneously with GS for all nonstore operation.

GFT which occurs simultaneously with GF for double precision stores.

For purposes of simplifying the explanations of the various "G" cycles and the OU sequences, all instructions have been categorized into 12 groups of instruction classes. When an instruction is shown having (R) or (X) in parenthesis, it is implied that a single mnemonic is used to represent a group of instructions which vary only by the register referred to. Usually, these are the A,Q,AQ, or X0-X7 registers.

12.2-contd

The instruction classes are:

1a) Normal load type instructions:

AD(R), ADL(R), AWC(R), ADE, SB(R), SBL(R), SWC(R), NEG, NEGL, CMP(R), CNA(R), AN(R), CAN(R), OR(R), TSX$_n$, RET, LD(R), LC(R), LDT, LDI, RMCM, LDE, FLD, DFLD, SZN, EA(R), RPT, RPL, RPD, RSW, FSZN, CMG, ADL, ER(R), RTD, LXL, RCU, RCCL.

1b) Slow store and read-alter-rewrite instructions:

AS(R), AOS, SS(R), ANS(R), ERS(R), STC2, STI, STC1, ORS(R), STAC.

1c) Single precision fast store instructions:

ST(R), STCA, STCQ, STBQ, STBA, SMIC, STZ, STE, FST, STT, SXL.

1d) Fast load instruction:

LREG

2a) Special fixed point compare instructions:

CMK, CWL

2b) Double precision store instructions:

STAQ, DFST, SMCM, LACL

2c) Safe store instruction:

SREG

12.2 - contd.

3) Shifts and gray-to-binary conversion instructions:

ARS, LRS, QRS, ALR, ARL, LLR, LRL, QLR, QRL, GTB, ALS, QLS, LLS.

4) Special floating point instructions:

FNO, FNEG

5) Normalized floating point multiply instructions:

FMP, DFMP

6) Fixed point multiply instructions:

MPF, MPY

7) Unnormalized floating point multiply instructions:

UFM, DUFM

8) Floating point compare instructions:

FCMP, DFCMP, FCMG, DFCMG

9) Normalized floating point addition and subtraction instructions:

FAD, DFAD, FSB, DFSB

10) Unnormalized floating point addition and subtraction instructions:

UFA, DUFA, UFS, DUFS

12.2 - contd

11a) Binary coded decimal instruction:

BCD

11b) Floating point divide instructions:

FDV, DFDV

11c) Fixed point divide instructions:

DVF, DIV

11d) Inverse floating point divide instructions:

FDI, DFDI

12) Non-OU instructions:

STP, STB, LBR$_n$,LDCF, CAM, SAM, SBR$_n$,SCU, STCD, TSS, TRA, TOV, TZE, TNZ, TMI, TPL, TRC, NOP, CIOC, TNC, TTF, TEO, TEU, XEC, XED, DIS, DRL, MME, ZOP, MME$_2$, MME$_3$, MME$_4$, ADB, LDB, SAZ, SDBR, EAB, EAP, LDBR, TSB.

All of these groups of instructions, except 12, cause a specific sequence of "G" cycles to be performed within the OU.

The material to follow will present the various "G" Cycles (12.2.1), the OU Timing Pulse Generator (12.2.2), and the OU Timing Sequences (12.2.3).

12.2.1 Basic "G" Cycles

12.2.1.1 GS-Setup or Start Cycle

Whenever a new instruction is to be processed by the OU, the first "G" cycle is always GS. GS starts on $ SET GS when a new operation code has been strobed into the O-register, and terminates when the first general operation strobe $ occurs. The GS cycle allows sufficient time for the operation code to be decoded and all necessary switches to be activated so that the first $ strobe will cause the correct data to be strobed into the appropriate registers. During GS the operation code in the O-register is transferred to the C-register (on $ START), and various indicators are reset as required.

The duration of the GS cycle may be considered to consist of two portions. The first portion is variable in time duration and is measured from $ SET GS when a new instruction is accepted to $DRDY when the operand for this instruction is accepted by the OU. The latter portion of GS is of medium time duration defined from $ START (approximately $DRDY time) to $\$_0$.

GS is set independently of the type of instruction being processed (see 12.2.3), and may overlap the GF cycles of the preceding operation for instruction classes other than 1c. To better define control variations in the generation of the GS cycle and also within the cycle, specific instruction sequences will be examined.

Case I: Startup or Operations Unit Not Busy State

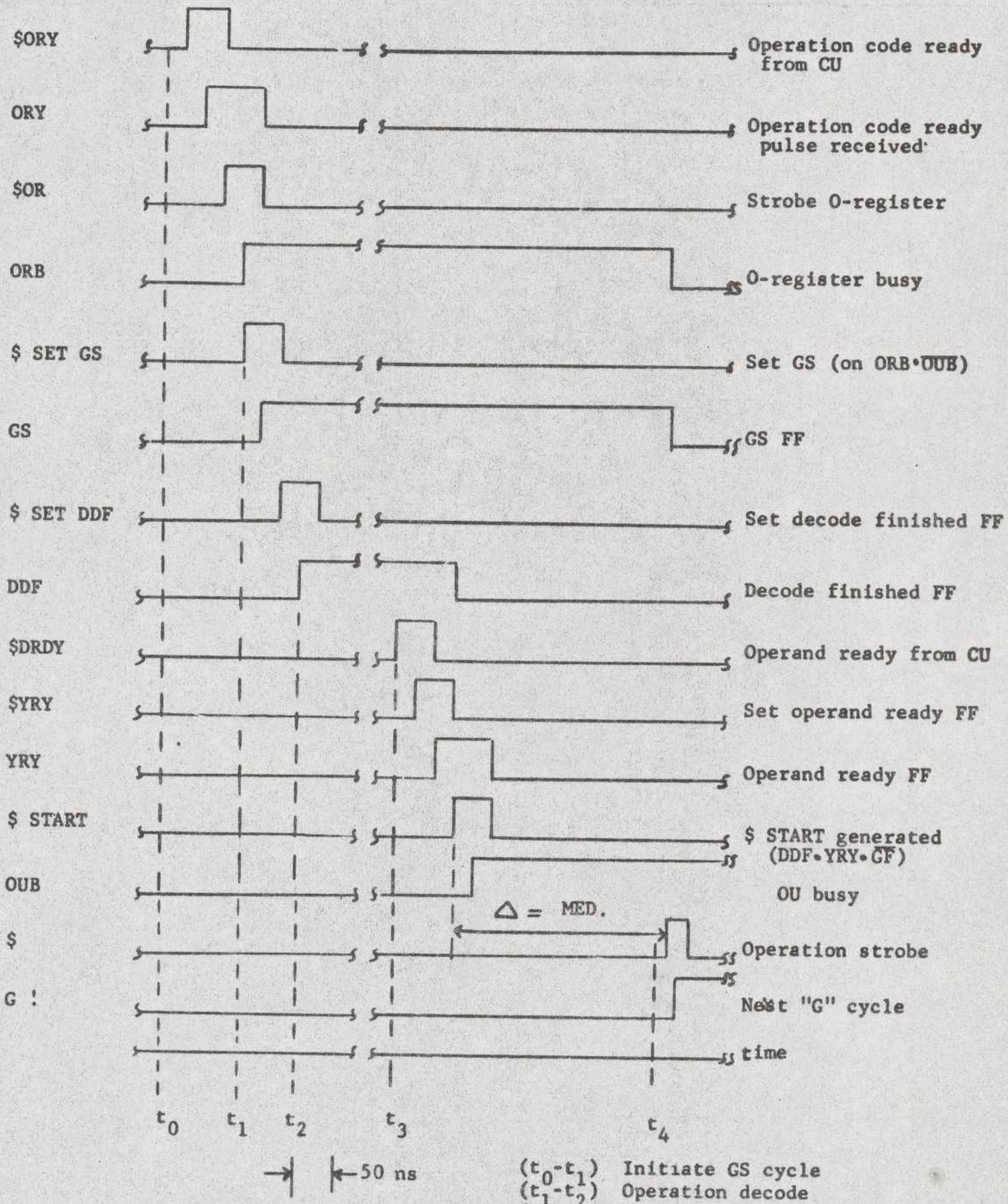
This state is defined by no OU timing cycle being in process and the OU being initialized from the Startup Fault or automatically initialized via the completion of a prior instruction.

Upon receipt of a $ ORY from the Control Unit indicating that the operation code for a new instruction is on the ZOR lines ready for processing the following sequence is initiated (refer to Fig. 12.2.1.1a).

1) $ ORY sets the ORY (Operation Code Ready) flip-flop.

2) The ORY flip-flop being set together with the ORB (0-register Busy) flip-flop being initialized (ORB = 0) will generate a pulse ($OR) which strobes the operation code on the ZOR lines into the 0-register.

   This pulse also sets the ORB flip-flop and resets the ORY flip-flop.


$ORY
Operation code ready from CU
ORY
Operation code ready pulse received
$OR
Strobe O-register
ORB
O-register busy
$ SET GS
Set GS (on ORB·$\overline{OUB}$)
GS
GS FF
$ SET DDF
Set decode finished FF
DDF
Decode finished FF
$DRDY
Operand ready from CU
$YRY
Set operand ready FF
YRY
Operand ready FF
$ START
$ START generated (DDF·YRY·$\overline{CF}$)
OUB
OU busy
$\triangle$ = MED.
$
Operation strobe
G !
Next "G" cycle
time
t_0
t_1
t_2
t_3
t_4
50 ns

(t_0-t_1) Initiate GS cycle
(t_1-t_2) Operation decode
(t_0-t_3) Minimum 350 ns
(t_2-t_4) Control point setup depending on decode.

Fig. 12.2.1.1a Basic Control of the "GS" Cycle from Startup or OU Not Busy State

12.2.1.1 - contd

3) The ORB flip-flop being set together with the OUB (Operations Unit Busy) flip-flop being initialized ($\overline{OUB} = 1$) will generate a $ SET GS pulse which sets the GS cycle flip-flop, initiating the first portion of the GS interval.

The $ SET GS pulse enters the DDF (Decode Delay) delay line exiting approximately 50 nanoseconds later. This delayed pulse ($ SET DDF) sets the DDF flip-flop which assures that the operation code contained in the O-register will have settled at individual control points sufficiently early with respect to the first pulse which will sample these points.

4) The GS flip-flop being set will gate the op code in the O-register to the output of the V switch. This switch output is the central distribution point for operation code to all control points for OU timing intervals other than GF.

5) The OU is notified by the Control Unit that the operand for the instruction is ready on the ZDI lines by $DRDY (Data Ready). This pulse will occur approximately 1 microsecond after $ORY and 350 nanoseconds after $ORY if the operand is a direct (no memory cycle required.)

This pulse results in the strobing of the operand on the ZDI lines into the M-register. $DRDY also generates $YRY which sets the YRY (Operand Ready) flip-flop signifying that the operand is in the M-register ready for processing.

12.2.1.1 - contd

6) The operation code being settled (DDF = 1), the operand being ready (YRY = 1) and the OU not in the process of finishing a prior instruction ($\overline{GF}$ = 1) results in the generation of $ START which terminates the first portion of GS and initiates the effective OU portion.

   $ START also sets the OUB flip-flop and strobes the contents of the 0-register into the C-register, the secondary operation code register.

   $ START enters the OU pulse generator delay network (paragraph 12.2.2). It will exit in approximately 320 nanoseconds for all instructions as $\$_0$, the first arithmetic pulse for the operation.

7) $\$_0$ resets the GS cycle and sets the next timing interval required by the operation.

   GS resetting gates the C-register into the V switch.

   $\$_0$ resets the ORB flip-flop freeing the 0-register to receive the operation code for the next instruction.

12.2.1.1 - contd

CASE II: Overlapped Pair of Nonstore Instructions (Fig. 12.2.1.1b)

To optimize execution time, the Control Unit normally pulls a pair of instructions when executing an instruction fetch. To take advantage of the availability of this pair, the Operations Unit may start processing the odd (second) instruction prior to completion of the even (first) instruction.

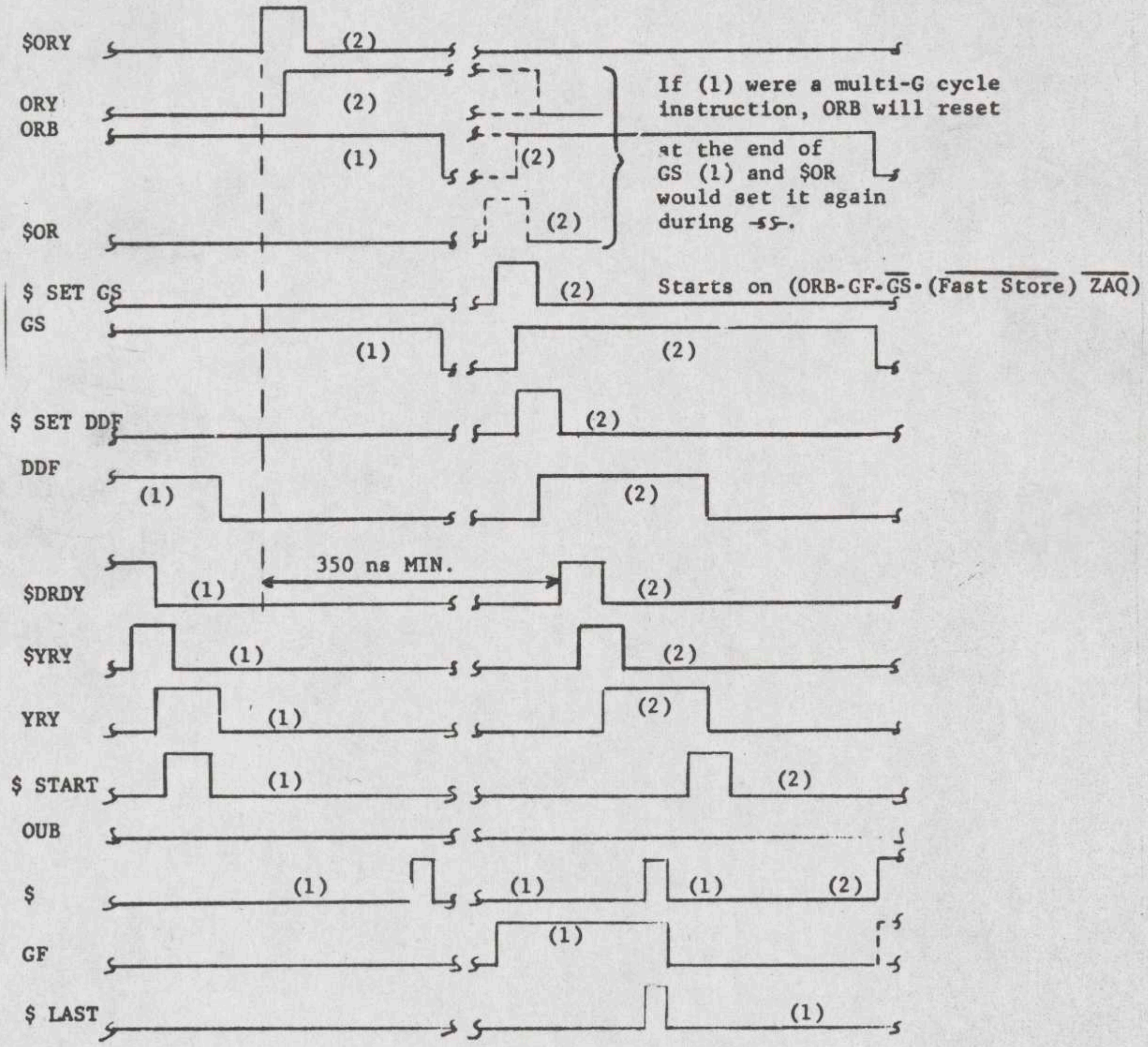
$ SET GS and $ START control for the even instruction is as explained in Case I. The following illustrates GS control for the odd instruction:

1) When $YRY is generated for the even instruction, it is returned to the Control Unit where upon receipt of this pulse the $ORY pulse for the odd instruction is generated and transmitted to the OU together with the operation code for the instruction.

2) $ORY sets the ORY flip-flop which was reset on the $ OR of the even instruction.

3) Upon completion of the GS cycle for the even instruction the ORB (0-register Busy) flip-flop is reset. Thus, as in Case I, $(\overline{ORB})$ (ORY) = 1 will generate a $ OR pulse which strobes the new operation code on the ZOR lines into the 0-register.

   $OR sets the ORB flip-flop and resets the ORY flip-flop.


$ORY
ORY
ORB
$OR
$ SET GS
GS
$ SET DDF
DDF
$DRDY
$YRY
YRY
$ START
OUB
$
GF
$ LAST
(1)
(2)
350 ns MIN.
If (1) were a multi-G cycle instruction, ORB will reset at the end of GS (1) and $OR would set it again during -ʃʃ-.
Starts on $(ORB \cdot GF \cdot \overline{GS} \cdot \overline{(\text{Fast Store})}\ \overline{ZAQ})$

NOTES:
1. (1) and (2) refer to overlapped instructions 1 and 2.
2. -ʃʃ- interval will be zero time for a GS-GF instruction, but will be variable for multi-G cycle instructions.
3. $YRY (1) initiates $ORY (2).
4. Under certain conditions $ LAST may be delayed.

Fig. 12.2.1.1b Basic Control of the "GS" Cycle for Nonstore Instructions with Instruction Overlap (Single Precision)

12.2.1.1 - contd

4) The condition which generated $ SET GS for the even instruction [(ORB) $(\overline{OUB})$] is inactive since the OUB flip-flop is still set and will remain set until completion of the even instruction. This term, then, will generate a $ SET GS only for Case I conditions (startup or no overlap).

$ SET GS for the odd instruction is generated only after the final (GF) cycle of the even instruction has been entered.

That is, $ SET GS = (ORB) (CF) $(\overline{GS})$

5) $ SET GS enters the DDF delay line exiting approximately 50 nanoseconds later. This delayed pulse ($ SET DDF) sets the DDF flip-flop.

6) When GS sets, the contents of the 0-register (containing the op code for the odd instruction) is gated to the output of the V switch. Control points affected by the odd instruction during GS are in the process of settling prior to completion of the even instruction. Decode for GF cycles is developed from the C-register; thus, there is no conflict resulting from the OU having two instructions active concurrently.

7) Approximately 300 nanoseconds after GS for the odd instruction has set, the final (GF) cycle of the even instruction will reset designating the completion of the instruction execution. The OUB flip-flop will not reset since (GF) $(\overline{GS})$ = 0.

—

12.2.1.1 - contd

8) Upon receipt of $DRDY (which normally occurs after completion of the previous instruction but can, in the case of a direct operand for the odd instruction or a multi-cycled even instruction, occur prior to termination of the even instruction) the operand for the odd instruction is strobed from the ZDI lines into the M-register. $DRDY generates $YRY which sets the YRY flip-flop indicating that the operand is ready for processing.

9) The operation code being settled (DDF = 1), the operand being ready (YRY = 1), and the even instruction being completed ($\overline{GF} = 1$), results in the generation of $ START for the odd instruction.

10) $ START strobes the contents of the 0-register into the C-register. $ START enters the OU pulse generator delay network (paragraph 12.2.2). It will exit in approximately 320 nanoseconds for all instructions as $\$_0$, the first arithmetic pulse for the operation.

11) $\$_0$ resets the GS cycle and sets the next timing interval required by the operation.

GS resetting gates the C-register into the V switch.

$\$_0$ resets the ORB flip-flop freeing the 0-register to receive the operation code for the next instruction.

12.2.1.2 GO - General Operate Cycle

The GO cycle is the main operate cycle for instructions which cannot be completed within a GS⟶GF sequence. The GO cycle is used when multiple arithmetic operations are required in order to execute the instruction.

For instruction classes 9 and 10, a single GO cycle is used to add the mantissas of the operands contained in the N-and A/Q-registers.

For instruction classes 3,5,6,7, and 11, multiple GO cycles are performed as explicitly implied by each instruction within these classes. Class 3 instructions require data shifting in which the contents of the N-register are shifted under control of the G counter. Class 5,6, and 7 instructions are multiply instructions which are performed under control of the G counter. To optimize the execution of multiply instructions a unique cycle, GOM (GO multiply) is utilized. GOM is generated in the RF frame where it is to be used which simplifies and speeds up the distribution of the function (GO) (MPLY). Class 11 instructions are Divide instructions which are performed under control of the G counter.

Instruction class 2a uses a single GO cycle for its execution. A logical AND is executed during the GO cycle of CMK. CWL uses a single GO cycle for an upper/lower limit comparison.

In classes 5,6,7, and 11, the multiple GO cycles are terminated with a final slow speed GO during which indicators are manipulated.

12.2.1.3 GC - Single Precision Store, or First Half of a Double Precision Store Cycle

GC is a special cycle which is operative during a single precision store (class 1c) or during the first half of a double precision store (class 2b). This is the only cycle whose duration is coincident with two other cycles. That is, GC = GS + GF for a single precision store or GC = GS + GT for a double precision store. For all other operations, GC = GS.

The GC cycle controls the gating of data registers which are to be stored into the Data Out Buffer although other timing cycles are moving during these instructions. Logic transients between these cycles will not be felt in the Data Out Buffer.

12.2.1.4 GT - Double Precision Store Operation Cycle

GT is operational for instruction classes 2b or 2c only and is set for the duration following the GS cycle of a double precision store operation until completion of the first half store of the operation. The duration of GT is T_M time dependent and always occurs between a GS and a GFT cycle. The GT cycle is terminated by a $DS from the Interface Frame.

12.2.1.5 GFT - Last Half of a Double Precision Store Cycle

GFT is a special cycle used only during the second half of a double precision store instruction (class 2b or 2c). GFT is always T_M time dependent and always follows a GT cycle.

12.2.1.6 GE - Floating Point Exponent Comparison Cycle

The GE cycle is used during the execution of instruction classes 8,9, or 10. The GE cycle is always a slow interval cycle and always follows GS cycle. During this cycle, two exponents are compared. The result of this comparison determines which exponent will be placed in the E-register, and which mantissas will be placed in the H- and N-registers. In addition, this comparison determines which mantissa, if either, requires alignment and how much, and whether the next cycle should be a GA, GO, or GF.

12.2.1.7 GA - Mantissa Alignment Cycle

The GA cycle is a fast interval cycle and always follows a GE cycle for instruction classes 8,9, and 10 if during that GE cycle, the exponent comparison had an absolute value of greater than zero and less than 72. During the GA cycle the smaller mantissa contained in the N-register will be shifted right 16, 4, or 1 bit(s) as determined by the G counter. If the G counter calls for more shifts GA will be repeated until LSA signals the last shift.

12.2.1.8 GD1 - First Divide Operation Cycle

A GD1 cycle will always follow a GS cycle of a class 11 instruction and will be a slow interval cycle.

For instruction classes 11a, 11b, and 11c the GD1 cycle defines the interval during which the divisor in M is transferred to H. The divisor is complemented if necessary to ensure a negative divisor. The N-register is cleared to zero and if the signs of the original dividend anddivisor are different the Quotient Sign flip-flop (QSF) is set.

12.2.1.8 - contd

For instruction class 11d the divisor in AQ is examined to determine if it is equal to zero. If divisor is zero, the Divide Check flip-flop (DVCK) will set and the OU will abort the operation. The dividend mantissa in M is transferred to H. The dividend exponent in M64-71 will be transferred to D. The N-register is cleared to zero and if the signs of the two original mantissas were different QSF will be set.

12.2.1.9 GD2 - Second Divide Operation Cycle

A GD2 cycle always follows a GD1 cycle and is a slow interval cycle.

For instruction classes 11a, 11b, and 11c the divisor in H is examined to see if it is equal to zero . If so, the Divide Check flip-flop (DVCK) is set and the OU will abort the operation. The positive dividend in AQ is transferred to N and the G counter is set for the appropriate number of GO cycles to execute the instruction.

For instruction class 11d the positive dividend in H is transferred to N, the negative divisor in AQ is transferred to H, the exponent registers D and E are exchanged, and the appropriate number of GO cycles to execute the instruction is loaded into the G counter.

12.2.1.10 GN - Mantissa Normalization and/or Overshift Correction Cycle

Although normalization is the primary function performed during GN, this cycle also may be entered for:

a) Compensating for overshifts of data when executing instructions in class 3. The compensations may be a right or left shift of one.

b) Backing out of an overflow condition which may have occurred while executing an instruction in classes 4, 5, 7, 9, or 10. This is accomplished with an arithmetic right shift of one with a corresponding increase in the exponent.

For instructions in classes 4,5,7,9, or 10 which enter GN for normalization, the mantissa to be normalized is in the N-register and is shifted left by 16,4, or 1 bit(s) at a time. The GN cycle will be repeated until normalization is complete. The G counter accumulates the number of normalization shifts.

12.2.1.11 GF - Final Cycle

GF is always the last "G" cycle to be performed within the OU when executing an instruction if no fault condition occurs earlier in the instruction execution. All final arithmetic operation, data transfers, indicator settings, etc. are performed during GF. This cycle is a slow interval for all nonstore operation, and a T_M interval for store operations.

12.2.2 Timing Pulse Generator

The purpose of the Timing Pulse Generator is to produce and distribute to the OU pulses ($) which are used to strobe all arithmetic and auxiliary registers in the OU. The $ strobe is a 50-nanosecond wide pulse, and the interval or period from the end of one strobe pulse to the end of the next strobe pulse is dependent upon the instruction which is being processed and the "G" cycle which is active at that particular time. The four intervals between $ strobes which are possible are:

1) Fast, which is approximately 200 nanoseconds.

2) Medium, which is approximately 320 nanoseconds.

3) Slow, which is approximately 400 nanoseconds.

4) T_M, which is memory speed dependent and is used to terminate all store data operations.

The three time intervals fast, medium, and slow have been selected in such a manner that any operation to be performed within the OU can be completed during one of these time periods. Below is an example of how one "G" cycle (GO) may be required by the operation being performed to be a fast, medium, or slow cycle.

Example:

A high speed GO would apply to shift cycles associated with shift operations. A medium speed GO would apply to iterative add cycles for multiply or divide. A slow speed GO would apply to operations which require setting of indicators at the end of the GO cycle as a result of arithmetic functions performed during the cycle.

12.2.2 - contd

The three nonmemory dependent time cycles are all obtained or developed by delay lines and logic arrangements. (See Fig. 12.2.2.) There are three delay lines which are cascaded via logic nets. When some condition occurs to initiate a timing cycle, a pulse starts down the first delay element. While this delay is still active, a decision is made as to whether this should be a fast or not a fast cycle. If the decision is for a fast cycle, the output of delay line is routed to the general $ distribution network and is inhibited from entering delay line 2; however, if the decision had been for a medium or slow cycle, then a pulse would have entered delay line 2 and would have been inhibited from entering the distribution network. After delay 2 has been propagated, and if this is to be a medium cycle, then the pulse is distributed to the system and inhibited from entering delay 3; however, if the cycle is to be slow then a pulse enters delay 3 and is inhibited from entering the $ distribution network. After delay 3 has been propagated, a slow cycle $ strobe is unconditionally distributed to the processor.

The first pulse to enter the delay chain for an instruction is $ START which is derived from the fact that a new operation designator has been set up in the OU, its operand is ready, and GF of a prior instruction is not active. The first delay path established will be a medium ($ START $\$_0$). Once the first $ strobe has been generated it becomes the input to delay line 1 to produce a subsequent $ strobe. Exceptions to the $ strobe being recirculated to produce later $ strobes are:

a) When store data operations are being performed, the $ strobe which starts the actual store operation may enter the delay line network but would not be enabled on the output so as to cause a $ strobe to be distributed to the system. For store operations, the next $ will be generated from the $DS line and completes the store operation. This is the T_M period mentioned earlier.

12.2.2 - contd

b) When the \$ strobe occurs which is the last \$ for the execution of an instruction (that is, the one which resets GF). GF being set inhibits this pulse from again entering the delay line network. This pulse generates \$ LAST. \$ LAST notifies the CU that an instruction execution has been completed.

c) When the OU is in Step mode. The Step mode flip-flop prevents any pulse from leaving the first delay line. In this mode each \$ pulse is generated as a function of the STEP switch.

Abbreviated equations describing the pulse generator are listed in terms of the "G" cycles and the OU instruction classes.

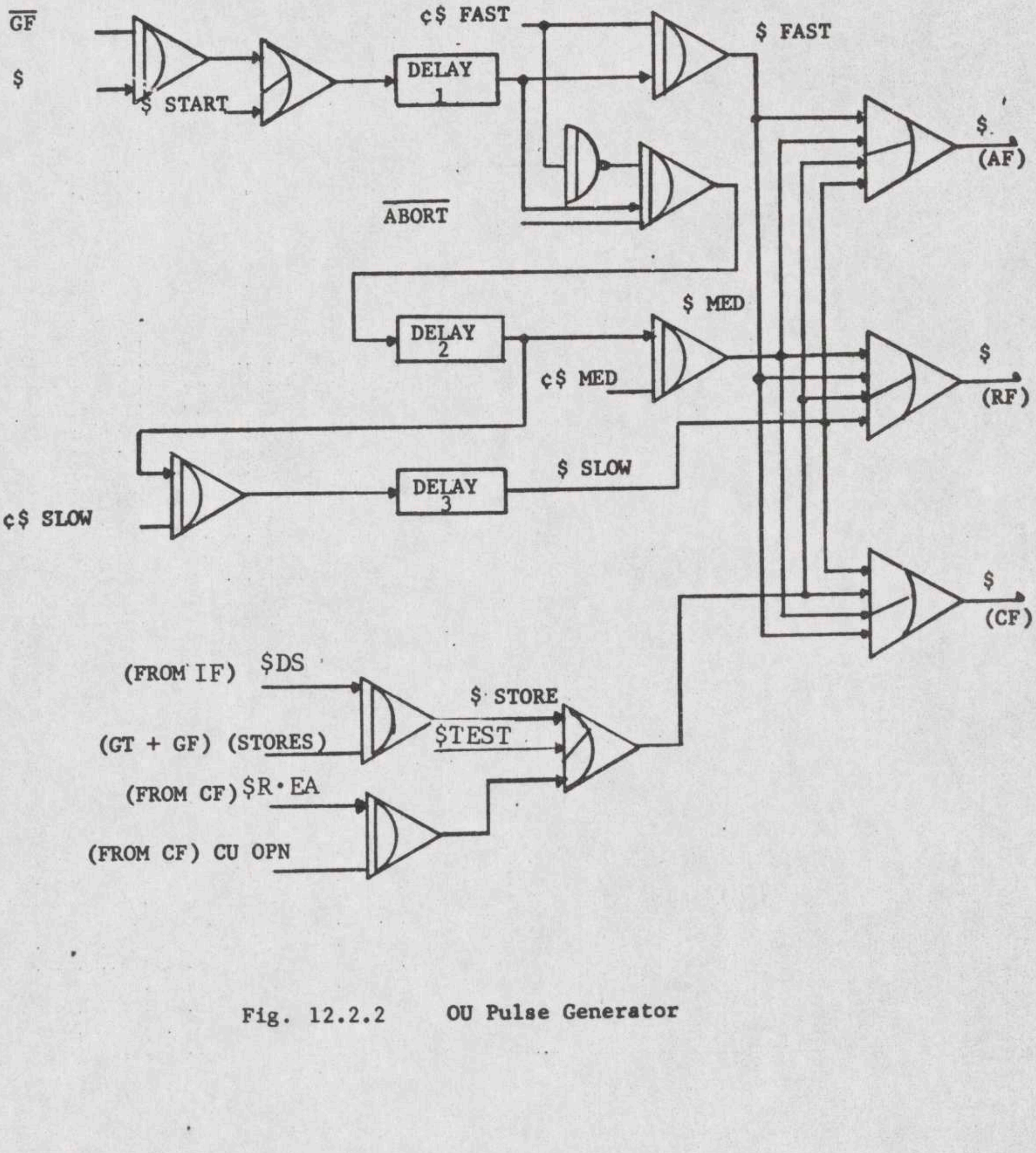
COMPANY CONFIDENTIAL

GENERAL ELECTRIC COMPANY
COMPUTER EQUIPMENT DEPARTMENT
PHOENIX, ARIZONA

Spec. No. M50EB00107

Cont. on Sh.12-26 Sh. No.12-25

TITLE: ENGINEERING PRODUCT SPECIFICATION
GE-645 PROTOTYPE PROCESSOR



$\overline{GF}$
$
$ START
¢$ FAST
DELAY 1
$ FAST
$ (AF)
$\overline{ABORT}$
DELAY 2
$ MED
¢$ MED
$ (RF)
¢$ SLOW
DELAY 3
$ SLOW
$ (CF)
(FROM IF) $DS
(GT + GF) (STORES)
$ STORE
$\overline{\$TEST}$
(FROM CF) $R·EA
(FROM CF) CU OPN

Fig. 12.2.2 OU Pulse Generator

12.2.2 - contd

Equations defining the enabling terms on the output gates of the fast, medium, and slow delay lines are:

$$\text{¢\$ FAST} = (\text{GO})\ (3) + \text{GN} + \text{GA}$$

$$\text{¢\$ MED} = (\text{GS}) + (\text{GO})\ [4 + (\text{GR}=1)\ (5 + 6 + 7 + 11)]$$

$$\begin{aligned}\text{¢\$ SLOW} = {} & (\text{GE}) + (\text{GD1}) + (\text{GD2}) + (\text{GF}) \\ & (1\text{b} + 1\text{c} + 2\text{b} + 2\text{c}) + (\text{GO}) \\ & [2\text{a} + 9 + 10 + (\text{GR} = 1)\ (5 + 6 + 7 + 11)] + \\ & \text{GO}\ (\text{CMFF})\end{aligned}$$

Equations defining the \$ output are:

$$\text{\$FAST} = [(\$)\ (\overline{\text{GF}}) + \text{\$ START}][\text{¢\$ FAST}][\Delta\ \text{FAST}]$$

$$\begin{aligned}\text{\$ MED} = {} & [\{(\$)\ (\overline{\text{GF}}) + \text{\$ START}\}(\overline{\text{¢\$ FAST}}) \\ & (\Delta\text{FAST})]\ (\text{¢\$ MED})\ (\Delta\text{MED})\end{aligned}$$

$$\begin{aligned}\text{\$SLOW} = {} & [\{(\$)\ (\overline{\text{GF}}) + \text{\$ START}\}(\overline{\text{¢\$ FAST}}) \\ & (\Delta\text{FAST})]\ (\Delta\text{MED})\ (\text{¢\$ SLOW})\ (\Delta\text{SLOW})\end{aligned}$$

12.2.2 - contd

$ STORE = ($DS) [GT + GF (1b + 1c + 2b +2c)]

$ SPECIAL*= $R·EA

The final output $ is:

$ = $ FAST + $ MED + $ SLOW + $ STORE + $R·EA + $ TEST

12.2.3 Timing Sequences

This paragraph defines the OU timing sequences which are used in the execution of instructions. This diagramatical representation reflects the various "G" cycles used and their time durations. The sequences shown are those which are necessary to execute all classes of instructions described in paragraph 12.2.

A detailed discussion as to what occurs during each sequence will be presented in paragraph 12.3.

---

*Resets OFL, EOF, and EUF Indicators on TOV, TEO, or TEU.

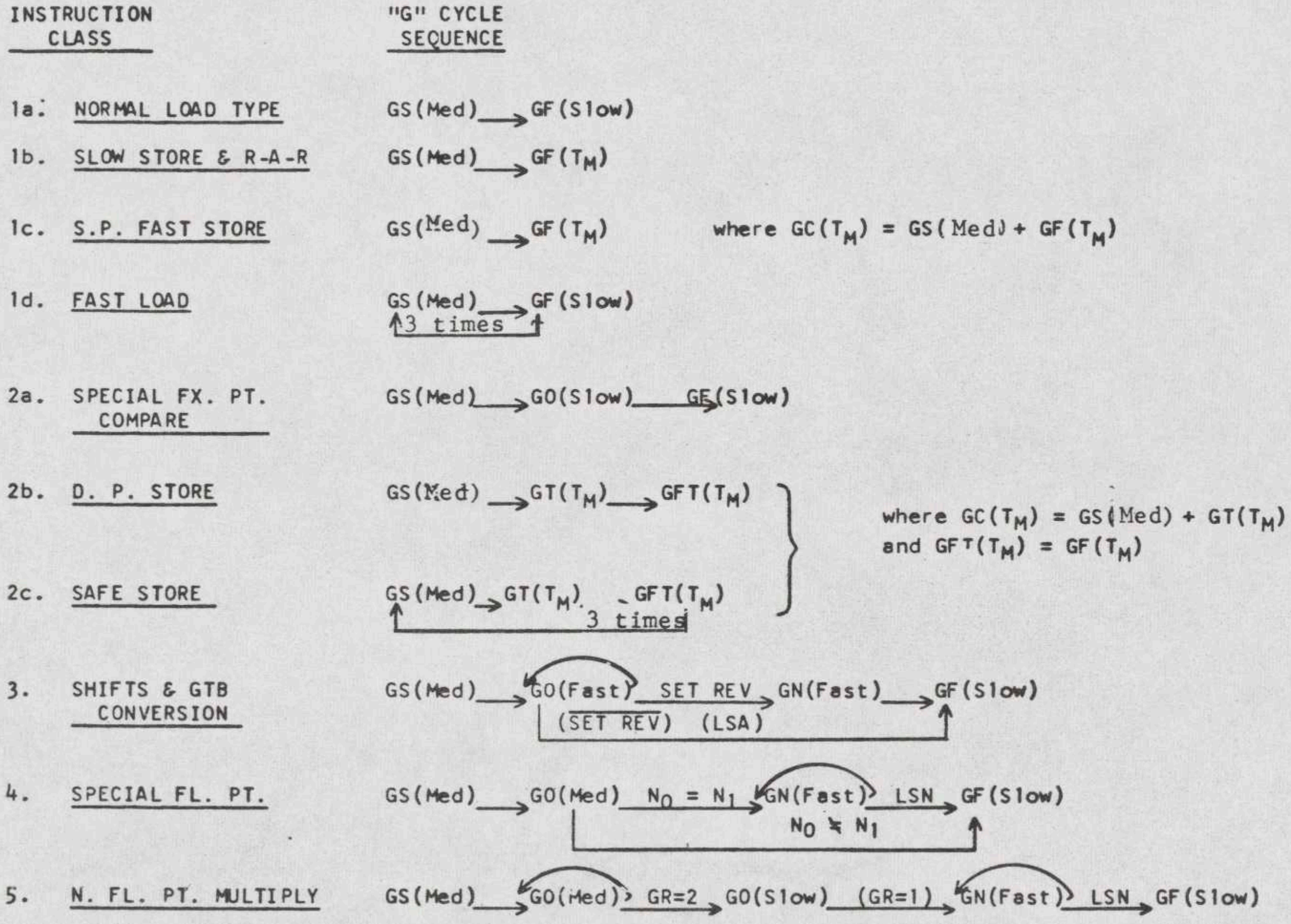
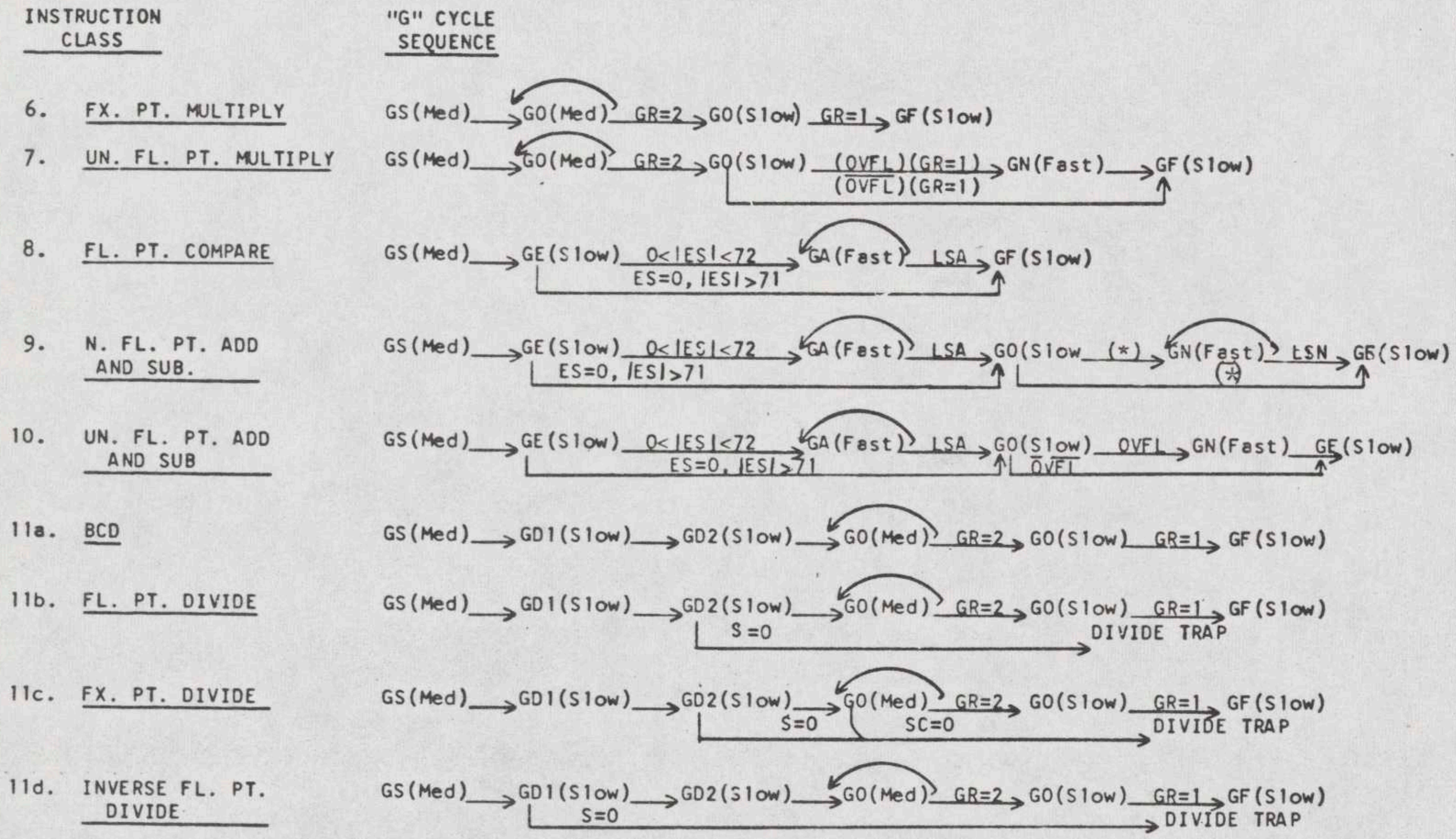

INSTRUCTION CLASS
"G" CYCLE SEQUENCE
1a. NORMAL LOAD TYPE
GS(Med) → GF(Slow)
1b. SLOW STORE & R-A-R
GS(Med) → GF(T_M)
1c. S.P. FAST STORE
GS(Med) → GF(T_M)
where GC(T_M) = GS(Med) + GF(T_M)
1d. FAST LOAD
GS(Med) → GF(Slow)
3 times
2a. SPECIAL FX. PT. COMPARE
GS(Med) → GO(Slow) → GF(Slow)
2b. D. P. STORE
GS(Med) → GT(T_M) → GFT(T_M)
where GC(T_M) = GS(Med) + GT(T_M)
and GFT(T_M) = GF(T_M)
2c. SAFE STORE
GS(Med) → GT(T_M) → GFT(T_M)
3 times
3. SHIFTS & GTB CONVERSION
GS(Med) → GO(Fast) SET REV → GN(Fast) → GF(Slow)
($\overline{\text{SET REV}}$) (LSA)
4. SPECIAL FL. PT.
GS(Med) → GO(Med) $N_0 = N_1$ → GN(Fast) LSN → GF(Slow)
$N_0 \neq N_1$
5. N. FL. PT. MULTIPLY
GS(Med) → GO(Med) GR=2 → GO(Slow) (GR=1) → GN(Fast) LSN → GF(Slow)

FIG. 12.2.3 OU Timing (G) Cycles
(Sheet 1 of 2)


INSTRUCTION CLASS
"G" CYCLE SEQUENCE
6. FX. PT. MULTIPLY
GS(Med) GO(Med) GR=2 GO(Slow) GR=1 GF(Slow)
7. UN. FL. PT. MULTIPLY
GS(Med) GO(Med) GR=2 GO(Slow) (OVFL)(GR=1) GN(Fast) GF(Slow)
(OVFL)(GR=1)
8. FL. PT. COMPARE
GS(Med) GE(Slow) 0<|ES|<72 GA(Fast) LSA GF(Slow)
ES=0, |ES|>71
9. N. FL. PT. ADD AND SUB.
GS(Med) GE(Slow) 0<|ES|<72 GA(Fast) LSA GO(Slow (*) GN(Fast) LSN GF(Slow)
ES=0, |ES|>71
(*)
10. UN. FL. PT. ADD AND SUB
GS(Med) GE(Slow) 0<|ES|<72 GA(Fast) LSA GO(Slow) OVFL GN(Fast) GF(Slow)
ES=0, |ES|>71
OVFL
11a. BCD
GS(Med) GD1(Slow) GD2(Slow) GO(Med) GR=2 GO(Slow) GR=1 GF(Slow)
11b. FL. PT. DIVIDE
GS(Med) GD1(Slow) GD2(Slow) GO(Med) GR=2 GO(Slow) GR=1 GF(Slow)
S=0
DIVIDE TRAP
11c. FX. PT. DIVIDE
GS(Med) GD1(Slow) GD2(Slow) GO(Med) GR=2 GO(Slow) GR=1 GF(Slow)
S=0
SC=0
DIVIDE TRAP
11d. INVERSE FL. PT. DIVIDE
GS(Med) GD1(Slow) GD2(Slow) GO(Med) GR=2 GO(Slow) GR=1 GF(Slow)
S=0
DIVIDE TRAP

* normalization required

FIG. 12.2.3 OU Timing (G) Cycles
(Sheet 2 of 2)

12.3 INSTRUCTION DESCRIPTION BY GROUP

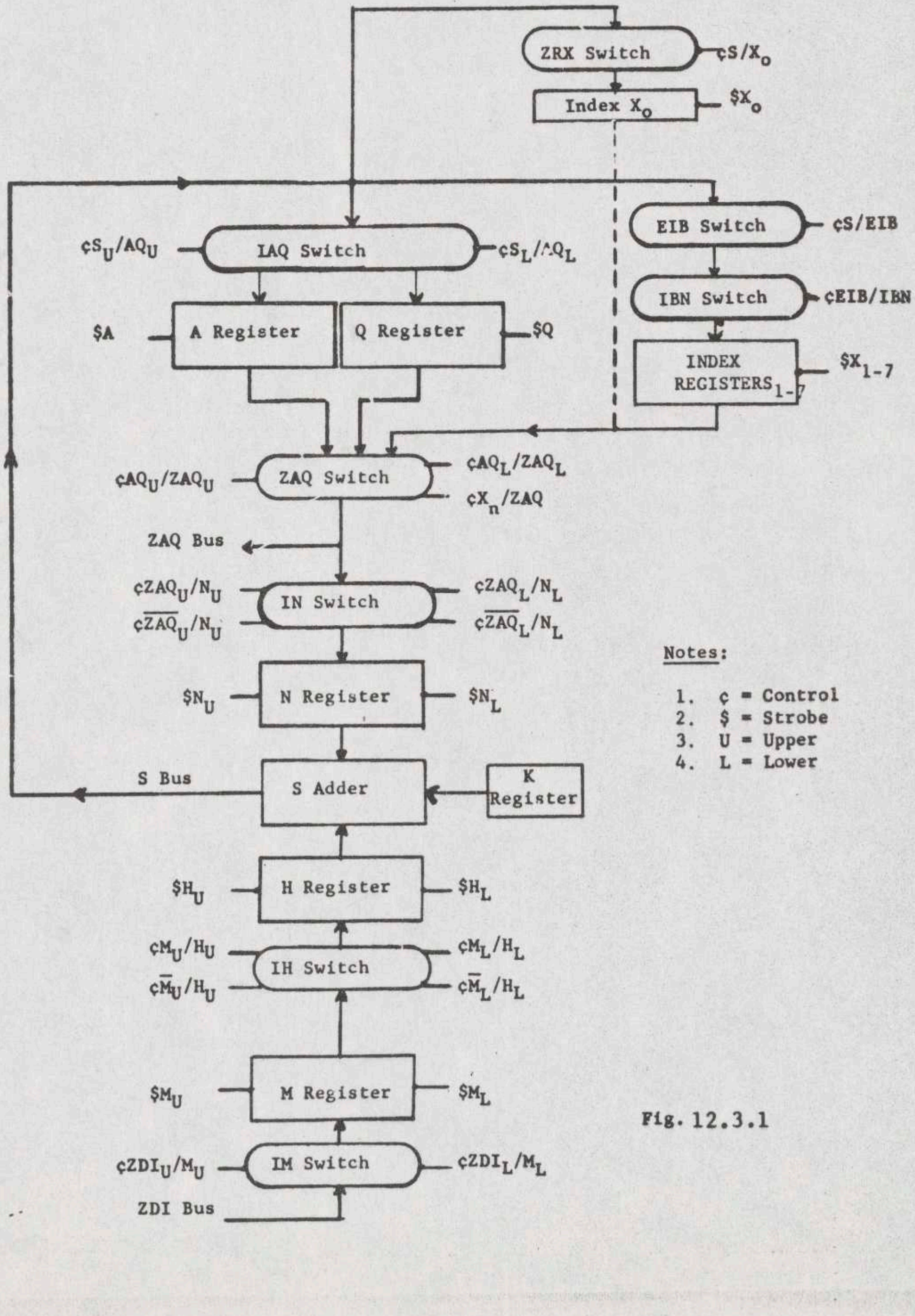
For purposes of describing the basic "G" sequences it was convenient to place all those instructions into a class which required a specific sequence for their execution. Generally, these sequences were independent of whether the instruction was fixed point or floating point, or single precision or double precision. However, for detailed explanations of instructions, (that is, data paths, control points, and timing differences between instruction within a class) the instructions are further subdivided to reflect fixed point or floating point and single precision or double precision. This subdivision is called a group.

12.3.1 FIXED POINT ADD GROUP

This group of instructions, which number approximately 174, constitute the majority of instructions executed in the OU. The Add Group includes a variety of fixed point instructions which require only two "G" cycles. The group consists of additions, subtractions, logicals, loads, and compares. The major data paths and control points for these instructions are shown in Fig. 12.3.1. In the execution of these instructions, adder registers H and N are loaded from some source, the S-adder performs an operation determined by the adder control flip-flops, K_1, K_2, K_3, and K_{72}, and the results are strobed to some location and/or examined for the purpose of setting Arithmetic Indicators. (For detailed description of the S-adder operation see 12.10.)

To illustrate the control used in the execution of these instructions, several instructions will be explained in detail. For the detailed control of all instructions in this group, refer to the OU Specification Charts 43D139962, page 2, 3, 5, and 11.

Major Data Paths and Control Points for the Fixed Point Add Type Instructions (OU)


ZRX Switch
¢S/X_o
Index X_o
\$$X_o$
EIB Switch
¢S/EIB
IBN Switch
¢EIB/IBN
INDEX REGISTERS$_{1-7}$
\$$X_{1-7}$
¢S_U/AQ_U
IAQ Switch
¢S_L/AQ_L
\$A
A Register
Q Register
\$Q
¢AQ_U/ZAQ_U
ZAQ Switch
¢AQ_L/ZAQ_L
¢X_n/ZAQ
ZAQ Bus
¢ZAQ_U/N_U
¢$\overline{ZAQ}_U/N_U$
IN Switch
¢ZAQ_L/N_L
¢$\overline{ZAQ}_L/N_L$
\$$N_U$
N Register
\$$N_L$
S Bus
S Adder
K Register
\$$H_U$
H Register
\$$H_L$
¢M_U/H_U
¢$\overline{M}_U/H_U$
IH Switch
¢M_L/H_L
¢$\overline{M}_L/H_L$
\$$M_U$
M Register
\$$M_L$
¢ZDI_U/M_U
IM Switch
¢ZDI_L/M_L
ZDI Bus

Notes:

1. ¢ = Control
2. \$ = Strobe
3. U = Upper
4. L = Lower

Fig. 12.3.1

12.3.1 - contd

In the explanation of all the example instructions, the following conditions will be assumed:

1) The instruction has been strobed in the O-register.

2) The GS cycle flip-flop has been set.

3) The DDF flip-flop has been set indicating the operation decode has settled.

NOTE: Reference to Fig. 12.3.1 and the partial timing diagram at the end of each example will aid in the understanding of the instruction execution and implementation.

12.3.1.1 Single Precision

The instructions comprising the single precision portion of the Add Group are:

ADLQ, ADLA, ADLX$_n$, AWCQ, AWCA, ADQ, ADA, ADX$_n$, CMPQ,

CMPA, CMPX$_n$, SBLQ, SBLA, SBLX$_n$, SWCQ, SWCA, SBQ, SBA,

SBX$_n$, CANQ, CANA, CANX$_n$, LCQ, LCA, LCX$_n$, ANQ, ANA, ANX$_n$,

CNAQ, CNAA, CNAX$_n$, LDQ, LDA, SZN, LDX$_n$, ORQ, ORA

ORX$_n$, RET, EAQ, LDT, EAA, LDI, EAX$_n$, ERQ, ERA, ERX$_n$,

TSX$_n$, LXL$_n$, NEG, CMG, ADL, RCU, RTD.

Example 1: Load (LDA)

During this instruction the contents of a memory location Y are loaded into the A-register via the S-adder. The presence of the LDA instruction code on the ZOR lines together with the Double Precision flip-flop (DPFF) reset, the Byte Control flip-flop (BCFF) reset, and the Multiply flip-flop (DIC) reset, enable the IM switch. The control points ¢ZDI$_{0-17}$/M$_{0-17}$, ¢ZDI$_{18-27}$/M$_{18-27}$, and ¢ZDI$_{28-35}$/M$_{28-35}$ are enabled presenting the operand lines (ZDI bus) to the M-register inputs. When the operand ready strobe ($DRDY) is received from the CU, the DPFF not being set allows the M-register strobe (M_{0-71}$) to be generated and strobe the LDA operand into the M-register. The $DRDY pulse is turned around as $YRY and returned to the CU to signal that a new instruction may be prepared. $YRY sets the operand ready flip-flop (YRY) in the OU. When YRY is set, the conditions exist to allow $ START to be generated by a last of three pulse detectors. ($ START) sets the OU Busy flip-flop (OUB), resets YRY and DDF, and transfers the operation code from the O-register to the C-register. It also enters the timing pulse generator delay line network.

12.3.1.1 - contd

When the $ START pulse propogates the medium delay elements, the operate strobe $ occurs. This allows the following events:

1) $H will strobe the LDA operand into the H-register from the M-register.

2) $N will strobe all zeros into the N-register since the IN switch is not enabled.

3) The Zero and Sign Indicators reset.

4) ORB to reset, indicating that the O-register is free to receive the operation designator for the next instructor.

5) GS to reset and GF to set.

The $ pulse will re-enter the timing pulse generator delay line network. GF being set will delay the pulse 400 nanoseconds before another $ pulse is generated.

During this 400 nanoseconds the contents of H and N are added together in the S-adder. The add function takes place because none of the K flip-flops are set. The addition of H (operand) and N (zeros) produces the operand at the S-adder outputs.

When GF set, the $¢S_{0-35}/AQ_{0-35}$ gate on the IAQ switch became enabled, making the S-adder outputs available on the inputs to the A-register. On the occurrence of the next $ strobe, $\$AQ_{0-35}$ is generated and the adder output is strobed into the A-register. At this time the contents of the memory location Y are loaded into the A-register. This $ strobe resets GF, turns on the Zero Indicator if the contents of A = 0, turns on the Sign Indicator if bit position 0 of A = 1, resets OUB if no other instruction has been initiated, and generates $ LAST. $ LAST is sent to the CU to indicate completion of the LDA instruction.

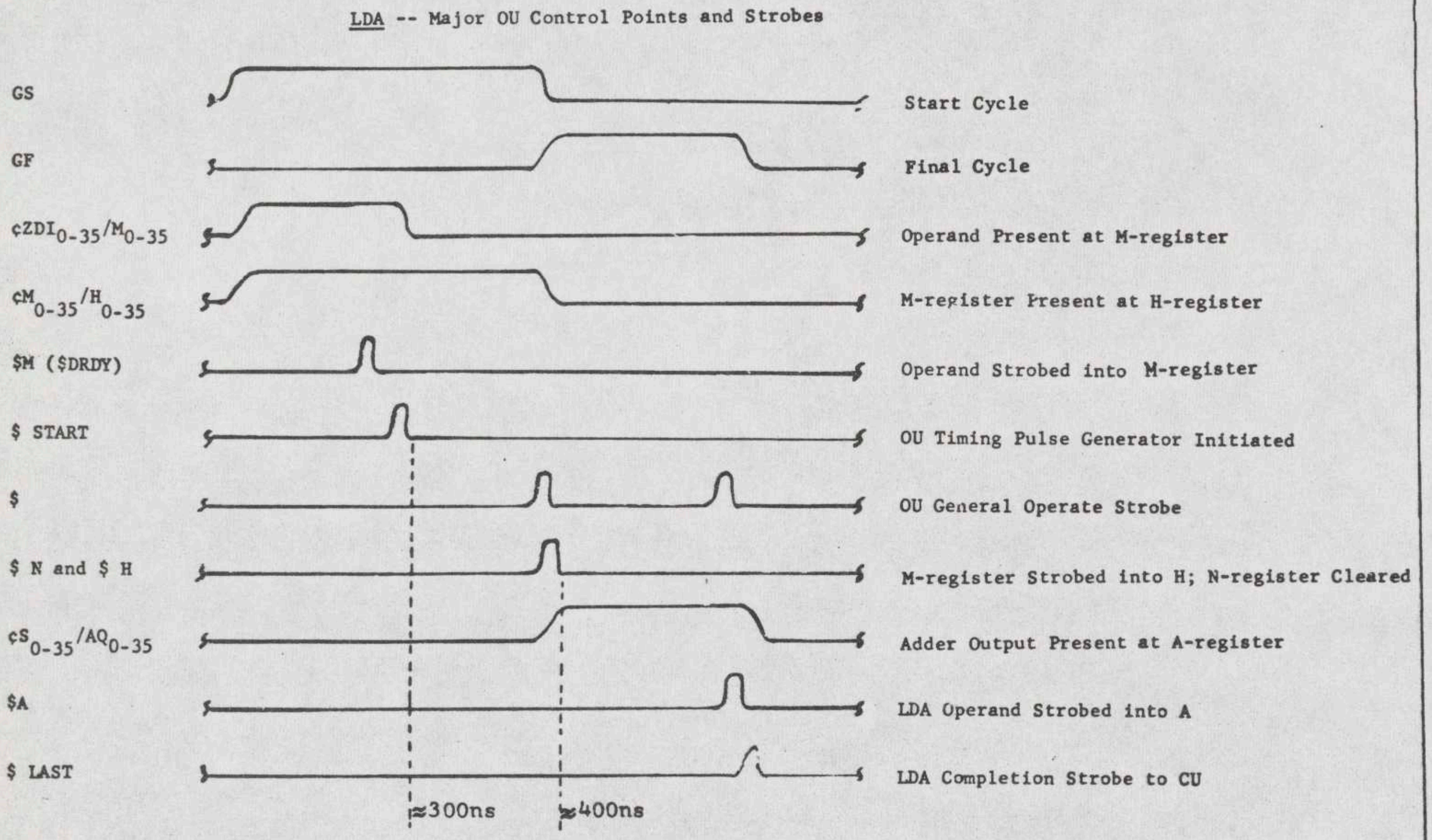

LDA -- Major OU Control Points and Strobes
GS
Start Cycle
GF
Final Cycle
¢ZDI_{0-35}/M_{0-35}
Operand Present at M-register
¢M_{0-35}/H_{0-35}
M-register Present at H-register
$M ($DRDY)
Operand Strobed into M-register
$ START
OU Timing Pulse Generator Initiated
$
OU General Operate Strobe
$ N and $ H
M-register Strobed into H; N-register Cleared
¢S_{0-35}/AQ_{0-35}
Adder Output Present at A-register
$A
LDA Operand Strobed into A
$ LAST
LDA Completion Strobe to CU
≈300ns
≈400ns

FIG. 12.3.1.1 LDA Instruction, Timing Diagram

12.3.1.1 - contd

Example 2: Add (ADX_n)

During this instruction the contents of bits 0-17 of a memory location Y are added to the contents of an index register X_n and the results are placed in X_n.

The presence of the ADX_n instruction code on the ZOR lines together with the DPFF reset, BCFF reset, and DIC reset enable the IM switch. The control point $¢ZDI_{0-17}/M_{0-17}$ is enabled to allow the operand lines (ZDI bus) to be present at the M-register inputs. When the operand ready strobe (\$DRDY) is received from the CU, DPFF not being set allows $\$M_{0-71}$ to be generated and strobe the ADX_n operand into the M-register. The \$DRDY pulse is turned around as \$YRY and returned to the CU to signal that a new instruction may be prepared. \$YRY sets the operand ready flip-flop (YRY) in the OU.

When YRY is set, the conditions exist to allow \$ START to be generated and enter the OU Timing Pulse Generator to later produce the operate strobe \$. \$ START also sets the OU busy flip-flop (OUB), resets YRY and DDF, and transfers the operation code from the O-register to the C-register.

GS

The IH switch is enabled by $¢M_{0-35}/H_{0-35}$, the ZAQ switch by $¢X_n/ZAQ_{0-17}$, and the IN switch by $¢ZAQ_{0-35}/N_{0-35}$. These enabled switches allow the operands from memory and the index register to be present at the inputs to the H- and N-register.

When the \$ START pulse propogates the medium delay elements, the operate strobe \$ occurs. This pulse:

1) Strobes (\$H) the memory operand from the M- to the H-register.

12.3.1.1 - contd

2) Strobes ($N) the present index register contents into the N-register.

3) Resets the Zero and Sign Indicators.

4) Resets the ORB flip-flop.

5) Resets the GS cycle and sets GF.

6) Re-enters the Timing Pulse Generator delay line network to produce another $ strobe in approximately 400 nanoseconds.

During this 400-nanosecond time the contents of the H- and N-registers are added in the S-adder. The Adder Control Register (K_1, K_2, K_3, K_{72}) = 0, 0, 0, 0 defines an add function which will be performed during the GF cycle.

GF

When GF set, control points on the inputs to all index registers were turned on. These control points are ¢S_{0-17}/ZRX_{0-17} for control of data into X_0, and ¢S_{0-17}/EIB_{0-17} and ¢EIB_{0-17}/IBN_{0-17} for control of data into X_{1-7}. On the occurrence of the next $ strobe, $$X_n$ is generated and the adder output [C(Y) + C(X_n)] is strobed into the index register specified by the instruction. The $ strobe also:

1) Resets GF.

2) Resets OUB if no other instruction has been initiated.

3) Sets the Sign Indicator if bit zero of the sum was negative ($S_0 = 1$).

Cont. on Sh.12-39

12.3.1.1 - contd

4) Sets the Zero Indicator if the sum was zero (S = 0).

5) Sets the Carry Indicator if a carry out of bit zero is generated during the add, ($SC_0 = 1$) or resets it if no carry ($SC_0 = 0$).

6) Sets the Overflow Indicator if the range of X_n is exceeded, (OFL = 1).

7) Generates $ LAST, which is sent to the CU to indicate completion of the ADX_n instruction.

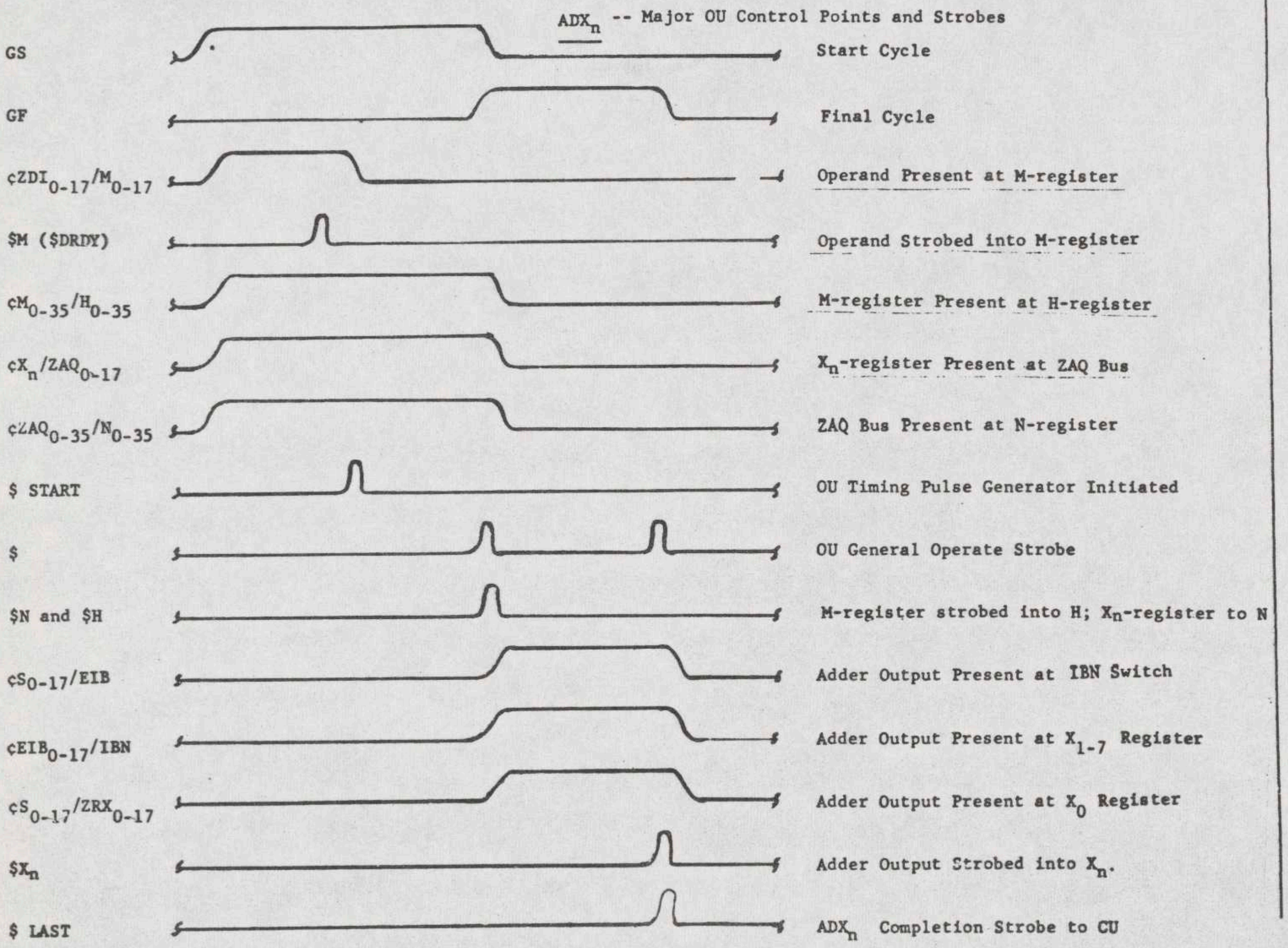

ADXn -- Major OU Control Points and Strobes
GS
Start Cycle
GF
Final Cycle
¢ZDI0-17/M0-17
Operand Present at M-register
$M ($DRDY)
Operand Strobed into M-register
¢M0-35/H0-35
M-register Present at H-register
¢Xn/ZAQ0-17
Xn-register Present at ZAQ Bus
¢ZAQ0-35/N0-35
ZAQ Bus Present at N-register
$ START
OU Timing Pulse Generator Initiated
$
OU General Operate Strobe
$N and $H
M-register strobed into H; Xn-register to N
¢S0-17/EIB
Adder Output Present at IBN Switch
¢EIB0-17/IBN
Adder Output Present at X1-7 Register
¢S0-17/ZRX0-17
Adder Output Present at X0 Register
$Xn
Adder Output Strobed into Xn.
$ LAST
ADXn Completion Strobe to CU

FIG. 12.3.1.1a ADXn Instruction, Timing Diagram

12.3.1.1 - contd

Example 3: Logical (ORA)

During this instruction the Boolean connective "OR" function is performed with the contents of the A-register and the contents of a memory location Y. The logical OR is performed in the S-adder and the results placed in the A-register.

The data path from ZDI/M and the generation of $ START for the ORA instruction is identical to that of the LDA instruction.

GS

During GS, the IH switch is enabled by ¢M_{0-35}/H_{0-35}, the ZAQ switch by ¢AQ_{0-35}/ZAQ_{0-35}, and the IN switch by ¢ZAQ_{0-35}/N_{0-35}. These enabled switches allow the operands from memory and the A-register to be present at the inputs to the H- and N-registers.

When the $ START pulse propogates the medium delay elements, the operate strobe $ occurs. This pulse permits the following:

1) The GS cycle to reset the GF cycle to set.

2) $H is generated which strobes the memory operand from the M- to the H-register.

3) $N is generated which strobes the present A-register contents into the N-register.

4) The Zero and Sign Indicators to reset.

5) The ORB flip-flop to reset.

6) Flip-flops K_1 and K_2 of the adder control register are set to place the adder in a Logical "OR" configuration.

12.3.1.1 - contd

7) The $ strobe enters the Timing Pulse Generator delay line network to produce another $ strobe in approximately 400 nanoseconds.

GF

During GF the Logical "OR" of the contents of the H- and N-registers is produced by the S-adder.

When the GF set, the ¢S_{0-35}/AQ_{0-35} gate on the IAQ switch became enabled making the S-adder outputs available on the inputs to the A-register. On the occurrence of the next $ strobe, $$AQ_{0-35}$ is generated and the adder output is strobed into the A-register. At this time the results of the Logical OR of a memory location Y and the A-register are placed in the A-register. This $ strobe also:

1) Resets the GF cycle flip-flop.

2) Resets the OUB flip-flop if no other instruction has been initiated.

3) Sets the Sign Indicator if bit zero of the "OR" resultant was a 1.

4) Sets the Zero Indicator if the "OR" resultant was zero.

5) Resets K_1 and K_2 adder controls.

6) Generates $ LAST, which is sent to the CU to indicate completion of the ORA instruction.

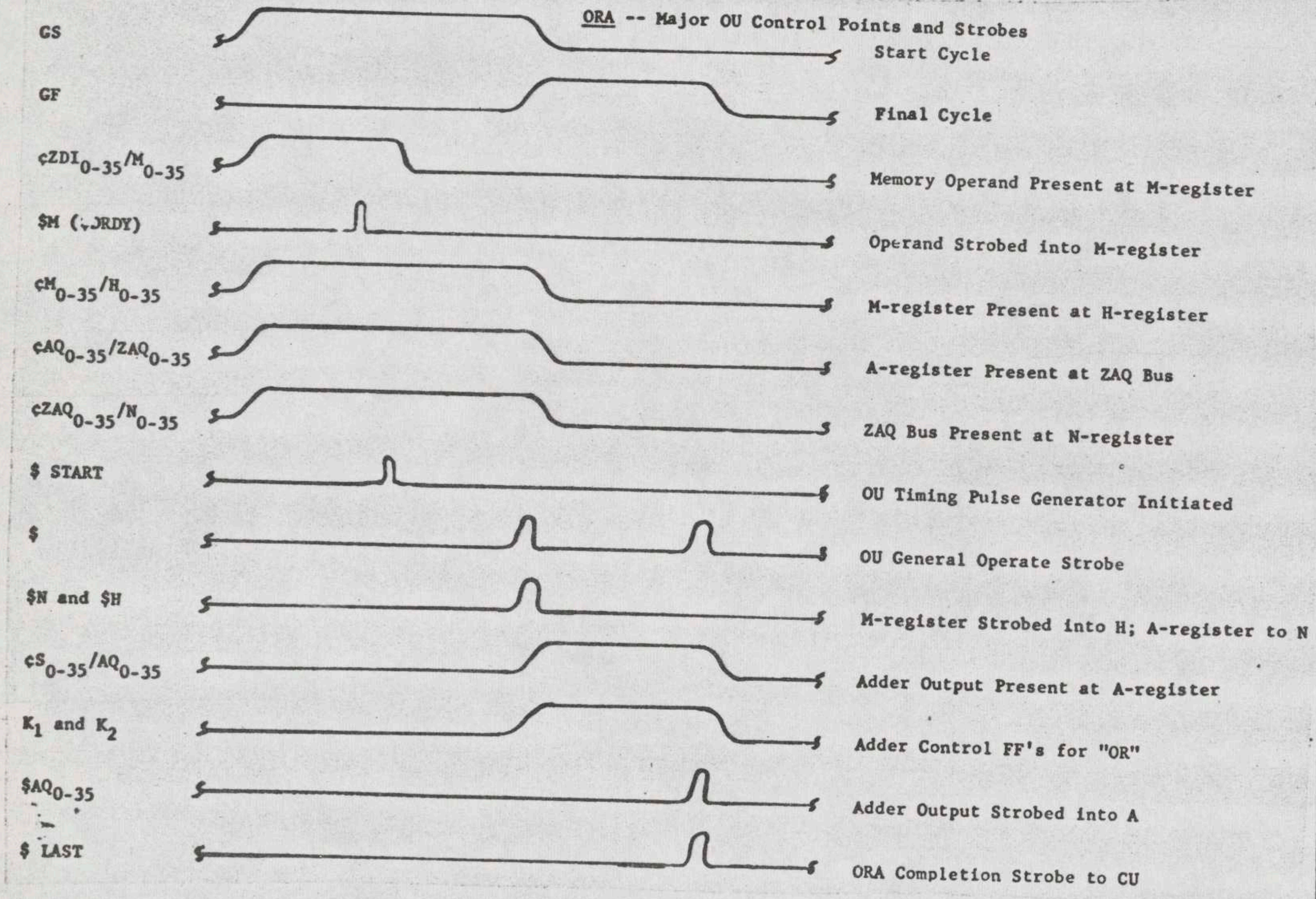

ORA -- Major OU Control Points and Strobes
GS
Start Cycle
GF
Final Cycle
¢ZDI$_{0-35}$/M$_{0-35}$
Memory Operand Present at M-register
$M (WORDY)
Operand Strobed into M-register
¢M$_{0-35}$/H$_{0-35}$
M-register Present at H-register
¢AQ$_{0-35}$/ZAQ$_{0-35}$
A-register Present at ZAQ Bus
¢ZAQ$_{0-35}$/N$_{0-35}$
ZAQ Bus Present at N-register
$ START
OU Timing Pulse Generator Initiated
$
OU General Operate Strobe
$N and $H
M-register Strobed into H; A-register to N
¢S$_{0-35}$/AQ$_{0-35}$
Adder Output Present at A-register
K$_1$ and K$_2$
Adder Control FF's for "OR"
AQ_{0-35}$
Adder Output Strobed into A
$ LAST
ORA Completion Strobe to CU

FIG. 12.3.1.1b ORA Instruction, Timing Diagram

ORA $(275)_8$
OR to Accumulator

All Registers are shown before the strobe at the end of the cycle.

GS

¢M_{0-35}/H_{0-35}

¢AQ_{0-35}/ZAQ_{0-35}

¢ZAQ_{0-35}/N_{0-35}

¢SET K_1, ¢SET K_2
¢RESET ZERO IND
¢RESET NEGATIVE IND
\$H \$N

Strobe on \$

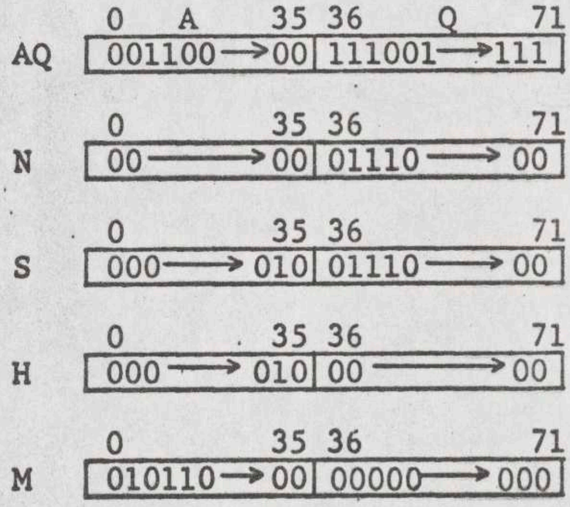

0
A
35
36
Q
71
AQ
001100
00
111001
111
0
35
36
71
N
00
00
01110
00
0
35
36
71
S
000
010
01110
00
0
35
36
71
H
000
010
00
00
0
35
36
71
M
010110
00
00000
000

FIG. 12.3.1.1c Register Contents During GS Cycle of ORA Instruction

---

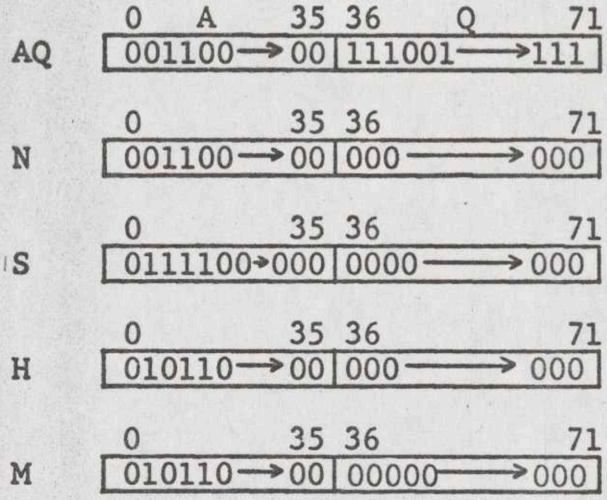
ORA $(275)_8$
OR to Accumulator - Cont

GF

¢S_{0-35}/A

¢ RESET K_1
¢ RESET K_2
¢ Set ZERO IND
  if S = 0
¢ SET NEGATIVE IND
  if S_0 = 1
$A

Strobe on $


0 A 35 36 Q 71
AQ 001100→00 111001→111
0 35 36 71
N 001100→00 000→000
0 35 36 71
S 0111100→000 0000→000
0 35 36 71
H 010110→00 000→000
0 35 36 71
M 010110→00 00000→000

Contents of Registers at the end of the "OR" operation. Note that the Q portion of the AQ is unaffected.

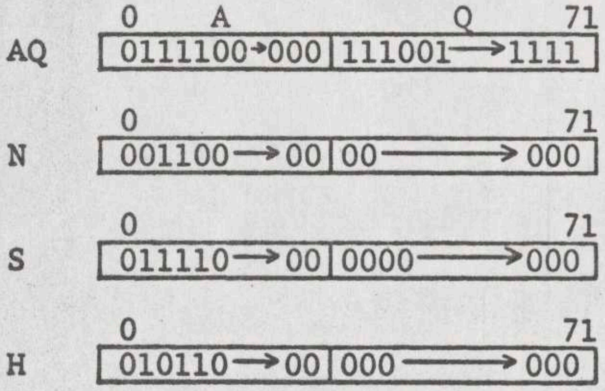

0 A Q 71
AQ 0111100→000 111001→1111
0 71
N 001100→00 00→000
0 71
S 011110→00 0000→000
0 71
H 010110→00 000→000

FIG. 12.3.1.1d Register Contents During GF Cycle of ORA Instruction

12.3.1.1 - contd

Example 4: Compare (CMG)

During this instruction the contents of a memory location Y are magnitude compared with the contents of the A-register and the Zero or Sign Indicators are set depending on the comparison. In general, a compare is a subtract operation where the register contents are not altered. Since this operation is a compare magnitude, the operands to be compared must be of unlike signs. If the numbers are of like signs, the memory operand must be two's complemented before the compare is actually made.

The data path from ZDI/M and the generation of $ START are the same as for the LDA instruction.

GS

During GS the signs of M_0 and A_0 are used to control the transfer of data from M to H. Because of the compare algorithm used, these signs must be opposite. Therefore, if the signs are not different, IH is enabled by ¢$\overline{M}_{0-35}/H_{0-35}$; if the signs are different, IH is enabled by ¢M_{0-35}/H_{0-35}. At the same time, the ZAQ switch is enabled by ¢AQ_{0-35}/ZAQ_{0-35} and the IN switch by ¢ZAQ_{0-35}/N_{0-35}. These enabled switches allow the operands from memory and the A-register to be present at the inputs to the H- and N-registers.

When the $ START pulse propogates the medium delay elements, the operate strobe $ occurs. This allows the following events:

1) The GS cycle to reset and the GF cycle to set.

2) $H is generated which strobes the memory operand from the M- to the H- register.

3) $N is generated which strobes the A-register contents into the N-register.

12.3.1.1- contd

4) If the signs of the A- and M-registers are the same, K_{72} control flip-flop will be set to force an initial carry into the adder.

5) The Zero and Sign Indicators are reset.

6) The ORB flip-flop to reset.

7) The $ strobe enters the Timing Pulse Generator delay line network to produce another $ strobe in approximately 400 nanoseconds.

GF

During GF the contents of the N- and H-registers are subtracted in the S-adder. (Refer to section 12.10 for a detailed description of the adder.)

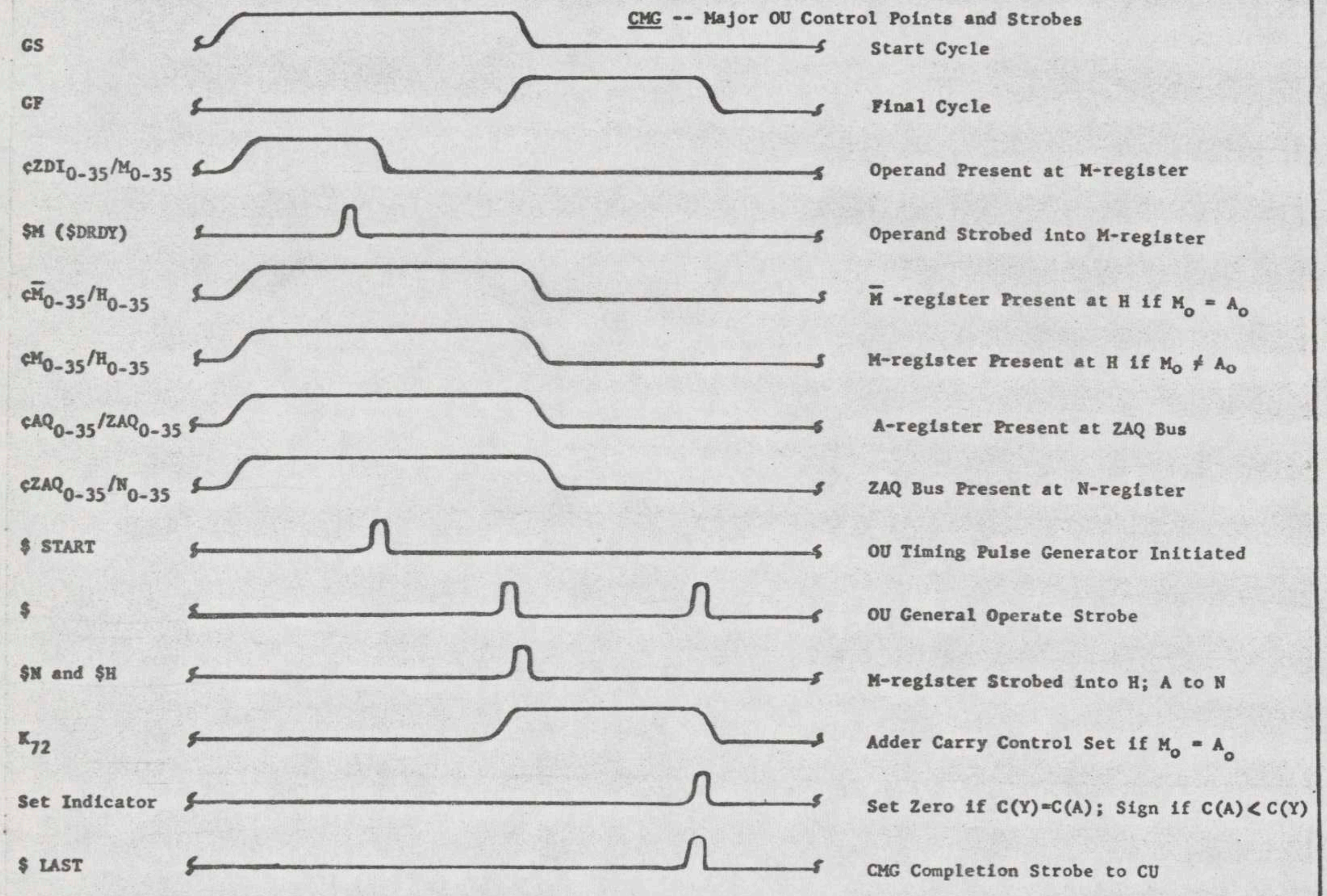
On the occurrence of the next $ strobe, an indicator may be set depending on the compare. If the contents of A and Y were equal, the Zero Indicator is set. If the contents of A were greater than Y, no indicator is set. If the contents of A were less than Y, the Sign Indicator is set. This $ strobe also:

1) Resets the GF cycle flip-flop.

2) Resets the OUB flip-flop if no other instruction has been initiated.

3) Resets the K_{72} control flip-flop if it is set.

4) Generates $ LAST, which is sent to the CU to indicate completion of the CMG instruction.



CMG -- Major OU Control Points and Strobes
GS
Start Cycle
GF
Final Cycle
$¢ZDI_{0-35}/M_{0-35}$
Operand Present at M-register
$M ($DRDY)
Operand Strobed into M-register
$¢\bar{M}_{0-35}/H_{0-35}$
$\bar{M}$ -register Present at H if $M_o = A_o$
$¢M_{0-35}/H_{0-35}$
M-register Present at H if $M_o \neq A_o$
$¢AQ_{0-35}/ZAQ_{0-35}$
A-register Present at ZAQ Bus
$¢ZAQ_{0-35}/N_{0-35}$
ZAQ Bus Present at N-register
$ START
OU Timing Pulse Generator Initiated
$
OU General Operate Strobe
$N and $H
M-register Strobed into H; A to N
K_{72}
Adder Carry Control Set if $M_o = A_o$
Set Indicator
Set Zero if C(Y)=C(A); Sign if C(A) < C(Y)
$ LAST
CMG Completion Strobe to CU

Fig 12.3.1.1E

FIG. 12.3.1.1e CMG Instruction, Timing Diagram

12.3.1.1 -contd

Example 5: Negate (NEG)

During this instruction, the contents of the A-register are converted to the negative of the original quantity by forming the two's complement of the number. The results are stored in the A-register.

No memory operand is required for this instruction. However, a $DRDY is still received from the CU for the necessity of generating $ START. The generation of this pulse is the same as for the instructions previously discussed.

GS

When the GS cycle is started and the NEG decode complete, the ZAQ switch becomes enabled by ¢AQ_{0-35}/ZAQ_{0-35}, and the IN switch by ¢$\overline{ZAQ}_{0-35}/N_{0-35}$ and ¢$\overline{ZAQ}_{36-71}/N_{36-71}$. These enabled switches allow the one's complement of the A-register to be present at the inputs of the N-register.

When the $ START pulse propogates the medium delay elements, the operate strobe $ occurs. This allows the following events:

1) The GS cycle to reset and the GF cycle to set.

2) $H is generated, but since the IH switch is not enabled all zeros are strobed into H.

3) $N is generated which strobes the one's complement of the A-register into N.

4) The Zero and Sign Indicators are reset.

5) The adder control flip-flop K_{72} is set to force the initial carry in forming the two's complement.

12.3.1.1 - contd

6) The ORB flip-flop is reset.

7) The $ strobe enters the Timing Pulse Generator to produce another $ strobe in approximately 400 nanoseconds.

GF

During GF the contents of H (H = 0) and N are added in the S-adder.

When GF set, the $¢S_{0-35}/AQ_{0-35}$ gate on the IAQ switch was enabled making the adder outputs available on the inputs to the A-register. On the occurrence of the next $ strobe, $\$AQ_{0-35}$ is generated and the adder output is strobed into the A-register. At this time the two's complement of the original contents of A are placed back in A. This $ strobe also:

1) Resets the GF cycle flip-flop.

2) Resets the OUB flip-flop if no other instruction has been initiated.

3) Sets the Sign Indicator if bit zero of the negated resultant is a one.

4) Sets the Zero Indicator if the negate resultant is zero.

5) Sets the Overflow Indicator if the range of A is exceeded.

6) Resets the K_{72} flip-flop.

7) Generates $ LAST, which is sent to the CU to indicate completion of the NEG instruction.

NEG -- Major OU Control Points and Strobes

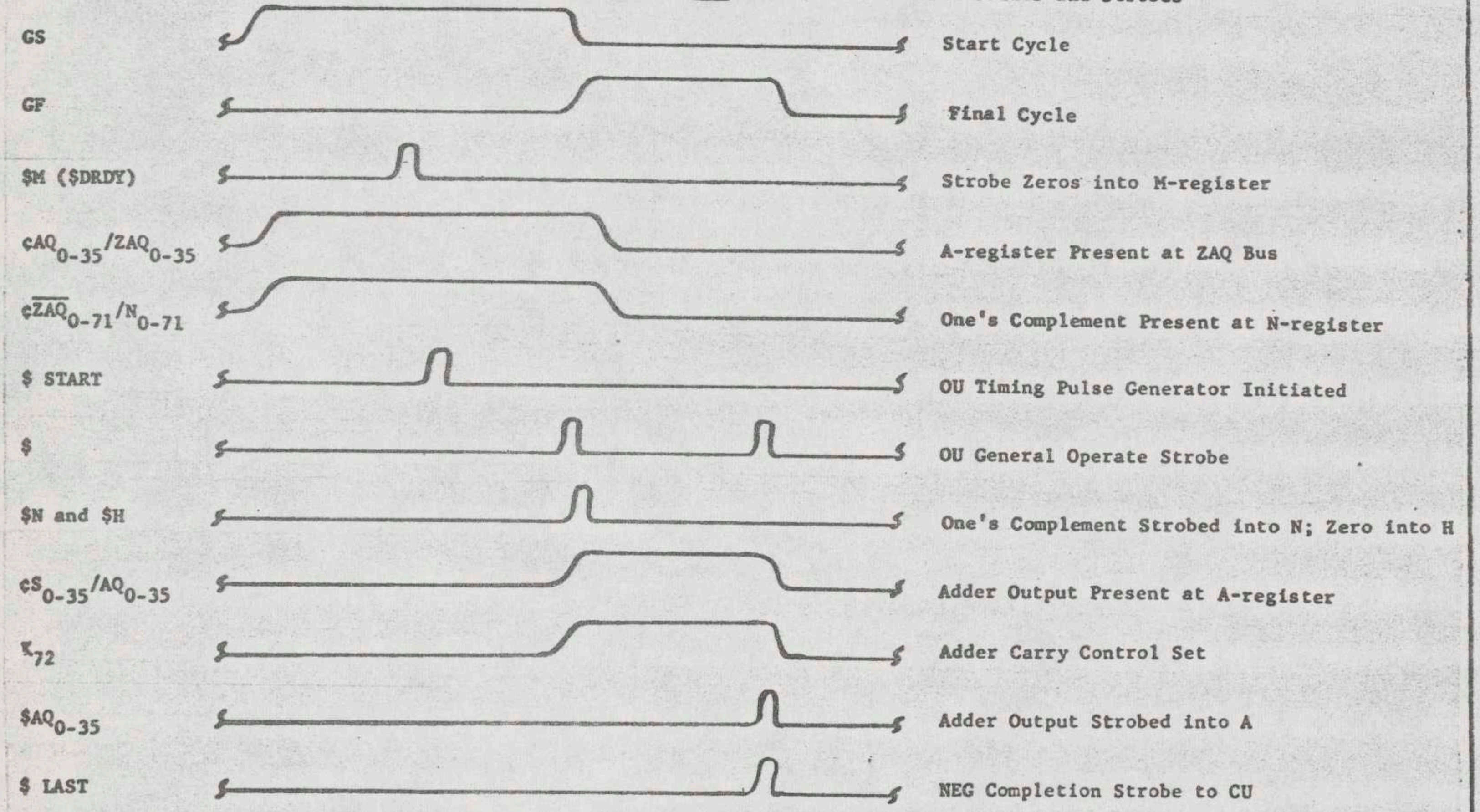

GS
Start Cycle
GF
Final Cycle
\$M (\$DRDY)
Strobe Zeros into M-register
¢AQ$_{0-35}$/ZAQ$_{0-35}$
A-register Present at ZAQ Bus
¢$\overline{ZAQ}_{0-71}$/N$_{0-71}$
One's Complement Present at N-register
\$ START
OU Timing Pulse Generator Initiated
\$
OU General Operate Strobe
\$N and \$H
One's Complement Strobed into N; Zero into H
¢S$_{0-35}$/AQ$_{0-35}$
Adder Output Present at A-register
K_{72}
Adder Carry Control Set
\$AQ$_{0-35}$
Adder Output Strobed into A
\$ LAST
NEG Completion Strobe to CU

FIG. 12.3.1.1f NEG Instruction, Timing Diagram

12.3.1.1 - contd

Example 6: Logical (ANA)

During this instruction the Boolean connective "AND" function is performed with the contents of the A-register and the contents of a memory location Y. The logical AND is performed in the S-adder and the results are placed in the A-register.

The data path from ZDI/M and the generation of $ START is identical to the LDA instruction.

Because of an internal peculiarity in the S-adder when executing a logical product or AND, it is necessary to load the one's complement of the memory operand into the H-register. This is made necessary because the adder equation for a logical AND is $(N_i)\ (\overline{H_i})$.

GS

During GS the IH switch is enabled by $\text{¢}\overline{M}_{0-35}/H_{0-35}$. The ZAQ witch is enabled by $\text{¢}AQ_{0-35}/ZAQ_{0-35}$ and the IN switch by $\text{¢}ZAQ_{0-35}/N_{0-35}$. These enabled switches allow the operands to be present at the H- and N-registers.

The lower half of $\overline{M}$ is enabled into H, but this is a don't care condition. Since the N-register will contain only zeros in the lower half, this is sufficient to inhibit any one's from being produced in the lower half of the adder output.

When the $ START pulse propogates the medium delay elements, the operate strobe $ occurs. This allows the following events:

1) The GS cycle to reset and the GF cycle to set.

12.3.1.1 - contd

2) \$H is generated which strobes the complement of the memory operand from the M- to the H-register.

3) \$N is generated which strobes the A-register contents into the N-register.

4) The Zero and Sign Indicators to reset.

5) Flip-flops K_1 and K_3 of the adder control register are set to place the adder in the Logical "AND" configuration.

6) The ORB flip-flop to reset.

7) The \$ Strobe enters the Timing Pulse Generator delay line network to produce another \$ strobe in approximately 400 nanoseconds.

GF

During the GF the Logical "AND" of the contents of the H- and N-registers is generated in the S-adder.

When GF set, the $¢S_{0-35}/AQ_{0-35}$ gate on the IAQ switch was enabled making the adder outputs available at the inputs to the A-register. On the occurrence of the next \$ strobe, $\$AQ_{0-35}$ is generated and the adder output is strobed into the A-register. This \$ strobe also:

1) Resets the GF cycle flip-flop.

2) Resets the OUB flip-flop if no other instruction has been initiated.

3) Sets the Zero Indicator if the "AND" resultant is zero.

12.3.1.1 - contd

4) Sets the Sign Indicator if bit zero of the "AND" resultant is a one.

5) Resets K_1 and K_3 flip-flops.

6) Generates $ LAST, which is sent to the CU to indicate completion of the ANA instruction.

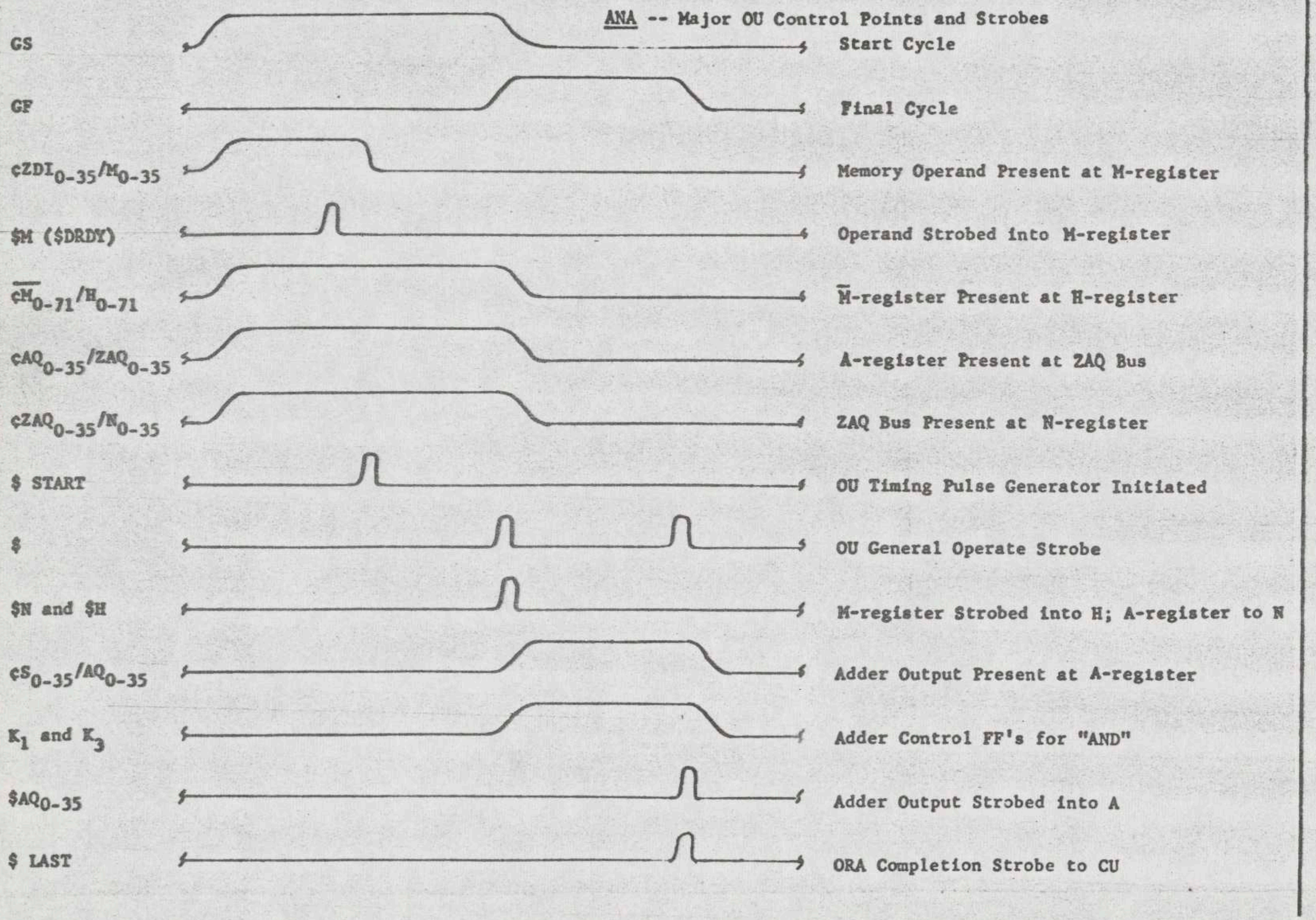

ANA -- Major OU Control Points and Strobes
GS
Start Cycle
GF
Final Cycle
$¢ZDI_{0-35}/M_{0-35}$
Memory Operand Present at M-register
$M ($DRDY)
Operand Strobed into M-register
$¢\overline{M}_{0-71}/H_{0-71}$
$\overline{M}$-register Present at H-register
$¢AQ_{0-35}/ZAQ_{0-35}$
A-register Present at ZAQ Bus
$¢ZAQ_{0-35}/N_{0-35}$
ZAQ Bus Present at N-register
$ START
OU Timing Pulse Generator Initiated
$
OU General Operate Strobe
$N and $H
M-register Strobed into H; A-register to N
$¢S_{0-35}/AQ_{0-35}$
Adder Output Present at A-register
K_1 and K_3
Adder Control FF's for "AND"
$\$AQ_{0-35}$
Adder Output Strobed into A
$ LAST
ORA Completion Strobe to CU

FIG. 12.3.1.1g ANA Instruction, Timing Diagram

ANA $(375)_8$
AND to Accumulator

All Registers are shown before the strobe at the end of the cycle.

GS

¢$\overline{M}_{0-35}/H_{0-35}$, ¢$\overline{M}_{36-71}/H_{36-71}$

¢AQ_{0-35}/ZAQ_{0-35}

¢ZAQ_{0-35}/N_{0-35}

¢SET K_1, ¢SET K_3
¢ RESET ZERO IND
¢ RESET NEGATIVE IND
\$H, \$N

} On \$

Register	0 – 35	36 – 71
AQ	AQ: 0101100 → 00	Q: 001111 → 111
N	1110111 → 1111	111 → 111
S	1111000 → 000	00 → 000
H	0000 → 000	000 → 001
M	010011 → 1111	000 → 0000

FIG. 12.3.1.1h Register Contents During GS Cycle of ANA Instruction

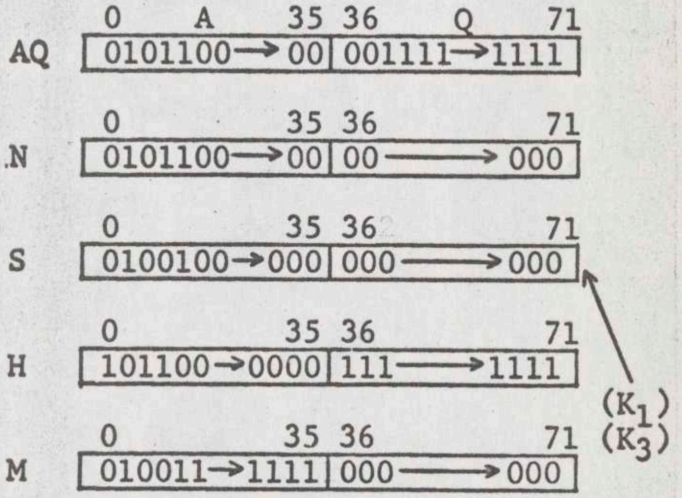
ANA $(375)_8$
AND to Accumulator - Contd

GF

¢S_{0-35}/A

¢ RESET K_1 ¢ RESET K_3
¢ SET ZERO IND if S = 0
¢ SET NEGATIVE IND if
$S_0 = 1$
on $

$A


0 A 35 36 Q 71
AQ 0101100→00 001111→1111
0 35 36 71
N 0101100→00 00→000
0 35 36 71
S 0100100→000 000→000
0 35 36 71
H 101100→0000 111→1111
(K1)
(K3)
0 35 36 71
M 010011→1111 000→000

Contents of Registers at the end of the ANA operation. Note that the Q portion of the AQ is unaffected.

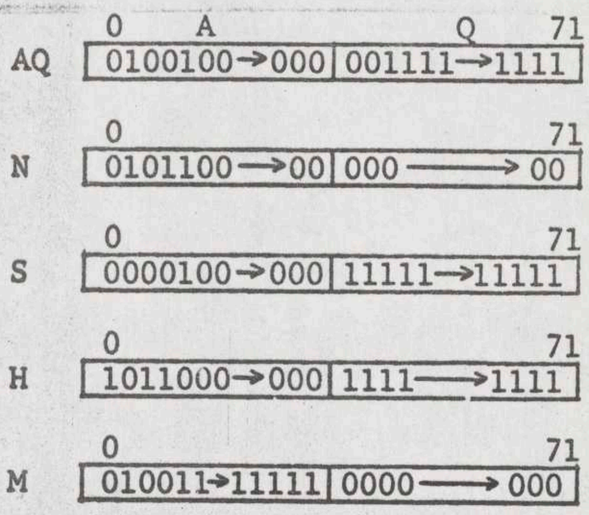

0 A Q 71
AQ 0100100→000 001111→1111
0 71
N 0101100→00 000→00
0 71
S 0000100→000 11111→11111
0 71
H 1011000→000 1111→1111
0 71
M 010011→11111 0000→000

FIG. 12.3.1.1i Register Contents During GF Cycle of ANA Instruction

12.3.1.1 - contd

Example 7: Add (ADL)

During this instruction a 72-bit operand is formed using a 36-bit word from memory by placing the memory word in the lower 36 through 71-bit positions of the H-register and forcing bits 0 through 35 to be the same as the sign bit of the memory word. When this operand is formed, it is added to the contents of the 72-bit AQ-register and the results are stored in AQ. This is the only instruction in the single precision portion of the Add Group which generates a 72-bit arithmetic resultant.

The control points ¢ZDI_{0-27}/M_{36-63} and ¢ZDI_{28-35}/M_{64-71} are enabled to allow the memory operand to be placed in the the lower half of the M-register on \$DRDY. \$ START is generated in the normal manner.

GS

During GS the IH switch is enabled by ¢M_{36-71}/H_{36-71} and ¢M_{0-35}/H_{0-35}. If bit zero of the memory operand (M_{36}) were a one, then ¢$\overline{M}_{0-35}$ is also enabled. The control of ¢M_{0-35}/H_{0-35} allows the upper half of the H-register to be filled with ones or zeros depending on the sign of the memory operand. The ZAQ switch is enabled by ¢AQ_{0-35}/ZAQ_{0-35} and ¢AQ_{36-71}/ZAQ_{36-71}. The IN switch is enabled by ¢ZAQ_{0-35}/N_{0-35} and ¢ZAQ_{36-71}/N_{36-71}. These enabled switches allow the generated 72-bit memory operand and the AQ operand to be present at the H- and N-registers.

When the \$ START pulse propogates the medium delay elements, the operate strobe \$ occurs. This allows the following events:

1) The GS cycle to reset and the GF cycle to set.

2) \$H is generated which strobes the memory operand plus its extended sign from the M--to the H-register.

12.3.1.1 - contd

3) \$N is generated which strobes the AQ-register contents into the N-register.

4) The Zero and Sign Indicators to reset.

5) The ORB flip-flop to reset.

6) The \$ strobe enters the Timing Pulse Generator delay line network to produce another \$ strobe in approximately 400 nanoseconds.

GF

During GF the contents of H and N are added in the S-adder.

When GF was set, the $¢S_{0-35}/AQ_{0-35}$ and $¢S_{36-71}/AQ_{36-71}$ gates on the IAQ switch were enabled making the adder outputs available at the inputs to the AQ-register. On the occurrence of the next \$ strobe, $\$AQ_{0-35}$ and a' $\$AQ_{36-71}$ are generated and the adder output is strobed into AQ. This \$ strobe also:

1) Resets the GF cycle flip-flop.

2) Resets the OUB flip-flop if no other instruction has been initiated.

3) Sets the Zero Indicator if the resultant of the ADL is zero.

4) Sets the Sign Indicator if bit zero of the ADL resultant is a one.

5) Sets the Overflow Indicator if the range of AQ is exceeded.

12.3.1.1 - contd

6) Sets the Carry Indicator if there was a carry out of bit zero of the adder; otherwise off.

7) Generates $ LAST, which is sent to the CU to indicate completion of the ADL instruction.

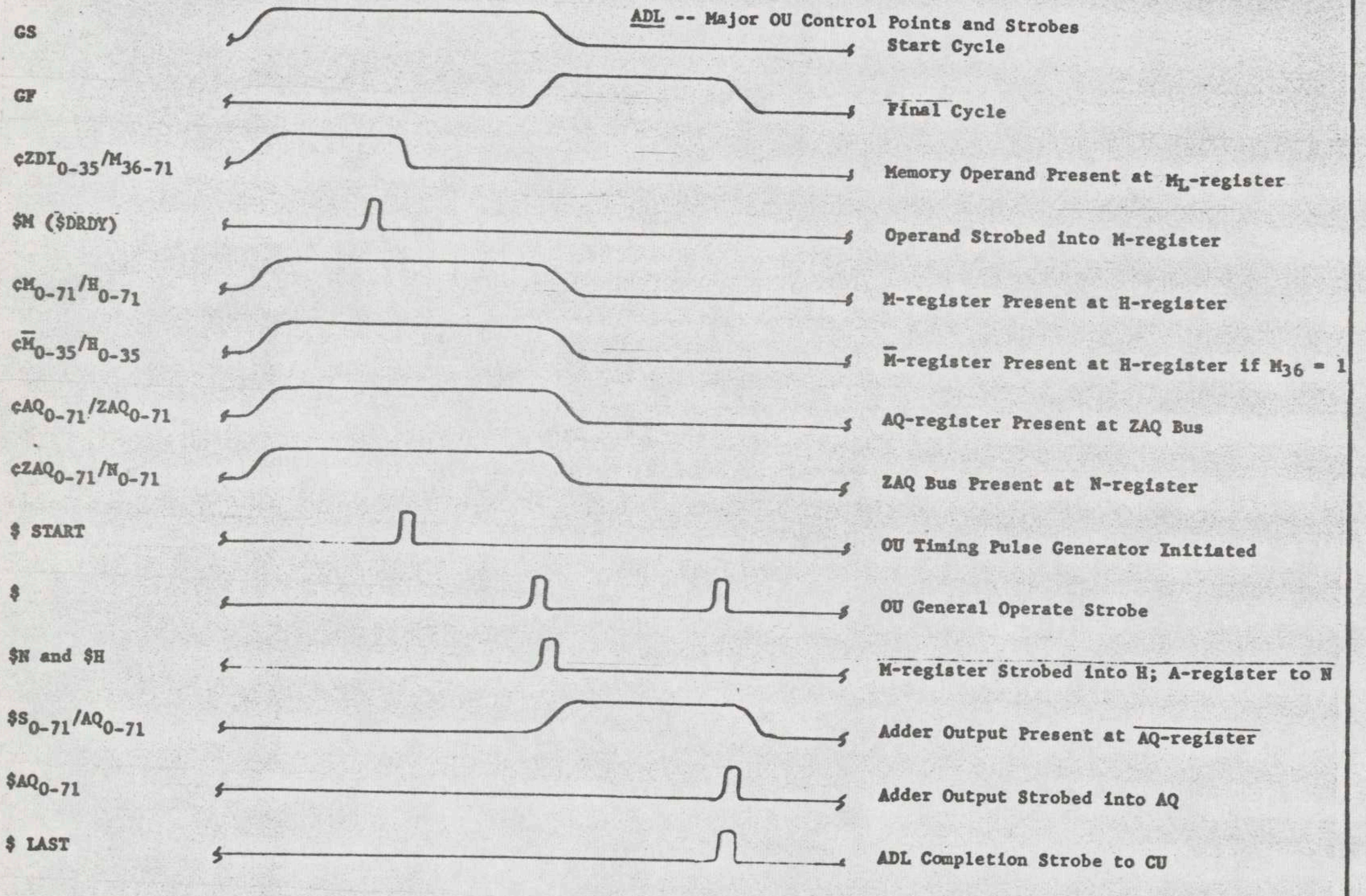

ADL -- Major OU Control Points and Strobes
GS
Start Cycle
GF
Final Cycle
¢ZDI$_{0-35}$/M$_{36-71}$
Memory Operand Present at M$_L$-register
$M ($DRDY)
Operand Strobed into M-register
¢M$_{0-71}$/H$_{0-71}$
M-register Present at H-register
¢$\overline{M}_{0-35}$/H$_{0-35}$
$\overline{M}$-register Present at H-register if M$_{36}$ = 1
¢AQ$_{0-71}$/ZAQ$_{0-71}$
AQ-register Present at ZAQ Bus
¢ZAQ$_{0-71}$/N$_{0-71}$
ZAQ Bus Present at N-register
$ START
OU Timing Pulse Generator Initiated
$
OU General Operate Strobe
$N and $H
M-register Strobed into H; A-register to N
S_{0-71}$/AQ$_{0-71}$
Adder Output Present at AQ-register
AQ_{0-71}$
Adder Output Strobed into AQ
$ LAST
ADL Completion Strobe to CU

FIG. 12.3.1.1j ADL Instruction, Timing Diagram

12.3.1.1 - contd

Example 8: Special Load RET - RTD - RCU.

All three of these instructions are treated the same by the OU. In fact the CF forces a 630 op code into the O-register for all three instructions. The only action of the OU is to load its Indicator Register from data supplied from memory. To determine what bit sets what indicator see GE-645 EPS, Sec. I.

These instructions are the same as the LDA (Example 1) except that during GF the contents of S_{18-27} are strobed directly into the Indicator Register.

12.3.1.2 Double Precision

The instructions comprising the double precision portion of the Add Group are:

RMCM, NEGL, ADLAQ, ADAQ, CMPAQ, SBLAQ, SBAQ, CANAQ, LCAQ, ANAQ, CNAAQ, LDAQ, ORAQ, ERAQ, RCCL.

The only difference in the OU between these instructions and those in the single precision portion of the Add Group is that these instructions manipulate 72 bits of data instead of 36, and two $DRDY's are required to accumulate input data in the M-register instead of one.

Example 1: Logical (CANAQ)

During this instruction, the contents of a pair of memory locations are compared logically with the contents of the AQ-register by means of the Boolean connective "AND" function. The only differences between this and the ANA instruction described in Section 12.3.1.1 is that this instruction is double precision, only sets indicators, and the contents of the AQ-register are not disturbed.

The IM switch is enabled by the presence of the CANAQ decode together with DPFF, BCFF, and DIC all reset. The control points ¢ZDI_{0-17}/M_{0-17}, ¢ZDI_{18-27}/M_{18-27}, and ¢ZDI_{28-35}/M_{28-35} are enabled to allow the ZDI bus to be present at the M-register inputs. When the first operand ready strobe $DRDY is received from the CU, the Double Precision flip-flop not yet being set allows the M-register strobe \$$M_{0-71}$ to be generated and strobe the most significant portion of the CANAQ operand into the M-register. This $DRDY strobe sets the Double Precision flip-flop (DPFF). The OU will not generate $ START until the second $DRDY is received. On the setting of DPFF, the IM switch becomes enabled by ¢ZDI_{0-27}/M_{36-63} and ¢ZDI_{28-35}/M_{64-71}.

12.3.1.2 - contd

When the second $DRDY is received about 300 nanoseconds after the first, the fact that DPFF is set and the CANAQ decode is true allows the M-register strobe $\$M_{36-71}$ to be generated and strobe the least significant portion of the CANAQ operand into the M-register. This $DRDY strobe resets DPFF and is turned around as $YRY and returned to the CU to signal that a new instruction may be prepared.

GS

During GS the IH switch is enabled by ¢$\overline{M}_{0-35}/H_{0-35}$ and ¢$\overline{M}_{36-71}/H_{36-71}$. The ZAQ switch is enabled by ¢AQ_{0-35}/ZAQ_{0-35} and ¢AQ_{36-71}/ZAQ_{36-71}, and the IN switch is enabled by ¢ZAQ_{0-71}/N_{0-71}. These enabled switches allow the operands to be present at the H- and N-registers.

When the $ START pulse propogates the medium delay elements, the operate strobe $ occurs. This allows the following events:

1) The GS cycle to reset and the GF cycle to set.

2) $H is generated which strobes the complement of the memory operand from the M- to the H-register.

3) $N is generated which strobes the AQ-register contents into N.

4) The Zero and Sign Indicators are reset.

5) Flip-flops K_1 and K_3 of the adder control register to set which place the adder in the Logical "AND" configuration.

6) The ORB flip-flop to reset.

12.3.1.2 - contd

7) The $ strobe enters the Timing Pulse Generator to produce another $ strobe in approximately 400 nanoseconds.

GF

During GF, the Logical "AND" of the contents of the H- and N-registers is produced in the S-adder.

When the next $ strobe occurs:

1) GF cycle flip-flop is reset.

2) OUB flip-flop is reset if no other instruction has been initiated.

3) The Zero Indicator is set if the "AND" results were zero.

4) The Sign Indicator is set if bit zero of the "AND" was a one.

5) Flip-flops K_1 and K_3 are reset.

6) Generates $ LAST, which is sent to the CU to indicate completion of the CANAQ instruction.

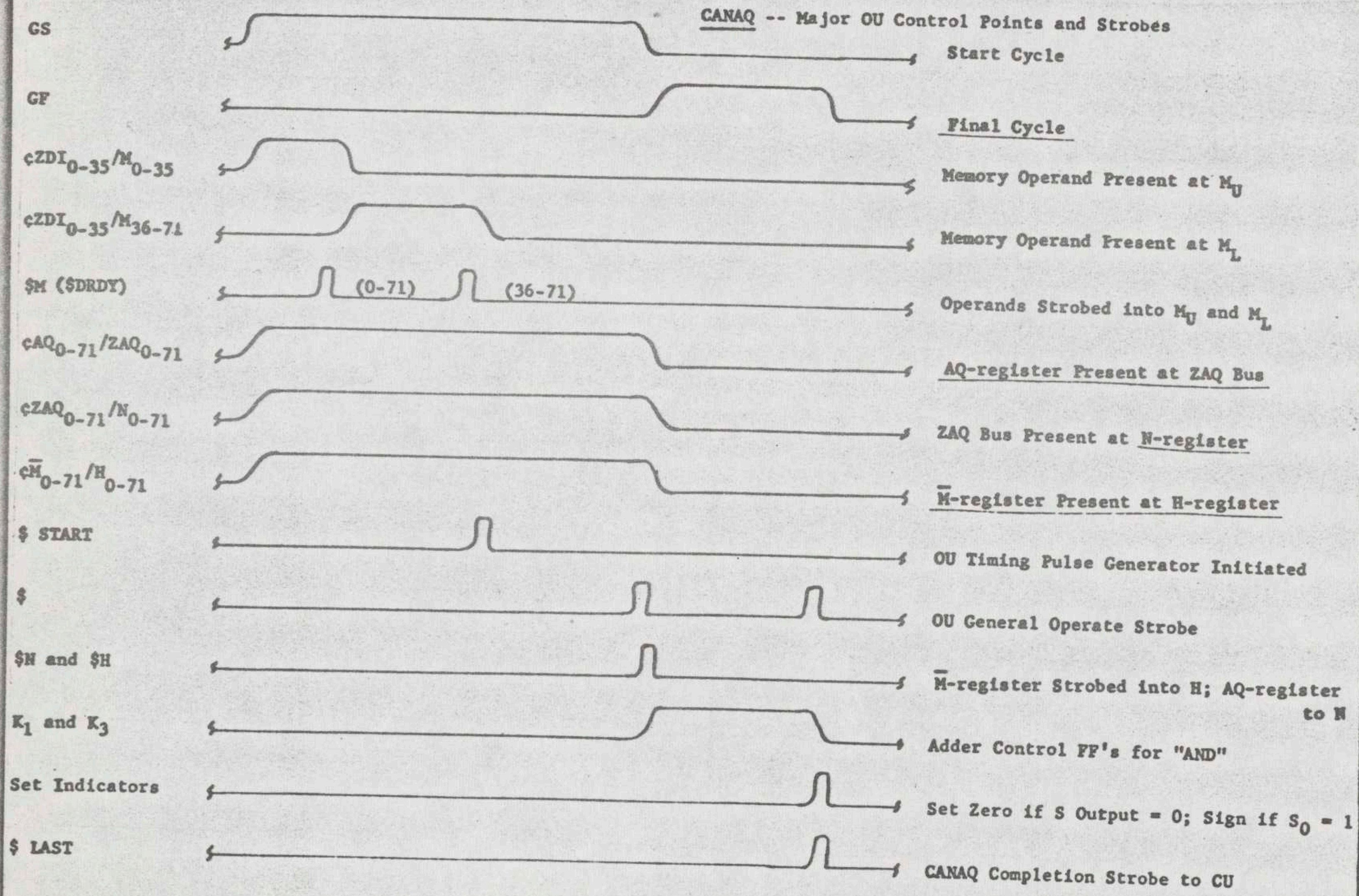

CANAQ -- Major OU Control Points and Strobes
GS
Start Cycle
GF
Final Cycle
¢ZDI$_{0-35}$/M$_{0-35}$
Memory Operand Present at M$_U$
¢ZDI$_{0-35}$/M$_{36-71}$
Memory Operand Present at M$_L$
$M ($DRDY)
(0-71)
(36-71)
Operands Strobed into M$_U$ and M$_L$
¢AQ$_{0-71}$/ZAQ$_{0-71}$
AQ-register Present at ZAQ Bus
¢ZAQ$_{0-71}$/N$_{0-71}$
ZAQ Bus Present at N-register
¢$\overline{M}_{0-71}$/H$_{0-71}$
$\overline{M}$-register Present at H-register
$ START
OU Timing Pulse Generator Initiated
$
OU General Operate Strobe
$H and $H
$\overline{M}$-register Strobed into H; AQ-register to N
K$_1$ and K$_3$
Adder Control FF's for "AND"
Set Indicators
Set Zero if S Output = 0; Sign if S$_0$ = 1
$ LAST
CANAQ Completion Strobe to CU

Fig 12.3.1.2

FIG. 12.3.1.2 CANAQ Instruction, Timing Diagram

12.3.1.2 - contd

Example 2: (SBAQ)

During this instruction the contents of a pair of memory locations are subtracted from AQ and the results are stored in AQ.

The IM switch is enabled by the presence of the SBAQ decode together with DPFF, BCFF, and DIC all reset. The control points ¢ZDI_{0-17}/M_{0-17}, ¢ZDI_{18-27}/M_{18-27}, and ¢ZDI_{28-35}/M_{28-35} are enabled to allow the ZDI bus to be present at the M-register inputs. When the first operand ready strobe $DRDY is received from the CU, the Double Precision flip-flop not yet being set allows the M-register strobe $\$M_{0-71}$ to be generated and strobe the most significant half of the SBAQ operand into the M-register. This $DRDY strobe sets the Double Precison flip-flop (DPFF). The OU will not generate $ START until the second $DRDY is received. On the setting of DPFF, the IM switch becomes enabled by ¢ZDI_{0-27}/M_{36-63} and ¢ZDI_{28-35}/M_{64-71}.

When the second $DRDY is received about 300 nanoseconds after the first, the fact that DPFF is set and the SBAQ decode is true allows the M-register strobe $\$M_{36-71}$ to be generated and strobe the least significant portion of the SBAQ operand into the M-register. This $DRDY strobe resets DPFF and is turned around as $YRY and returned to the CU to signal that a new instruction may be prepared.

GS

During GS the IH switch is enabled by $\overline{¢M}_{0-35}/H_{0-35}$ and ¢M_{36-71}/H_{36-71}. The bar side of the M-register is enabled into H because subtraction in the two's complement system requires that one operand be negated (one's complement) and the initial carry be forced.

12.3.1.2 - contd

The ZAQ switch is enabled by $¢AQ_{0-35}/ZAQ_{0-35}$ and $¢AQ_{36-71}/ZAQ_{36-71}$, and the IN switch by $¢ZAQ_{0-35}/N_{0-35}$ and ZAQ_{36-71}/N_{36-71}. These enabled switches allow the operands to be present at the H- and N-registers.

When the $ START pulse propogates the medium delay elements, the operate strobe $ occurs. This allows the following events:

1) The GS cycle to reset and the GF cycle to set.

2) $H is generated which strobes the complement of the memory operand from the M- to the H-register.

3) $N is generated which strobes the AQ-register contents into N.

4) The Zero and Sign Indicators are reset.

5) Flip-flop K_{72} is set to force the initial carry into the adder.

6) The ORB flip-flop is reset.

7) The $ strobe enters the Timing Pulse Generator delay line network to produce another $ strobe in approximately 400 nanoseconds.

GF

During GF the contents of H and N and subtracted in the adder.

When GF set, the IAQ switch was enabled by $¢S_{0-35}/AQ_{0-35}$ and $¢S_{36-71}/AQ_{36-71}$. This allows the adder outputs to be available at the inputs to the AQ-register.

12.3.1.2 - contd

When the next \$ strobe occurs, $\$AQ_{0-35}$ and $\$AQ_{36-71}$ are produced to strobe the adder output into AQ. This \$ strobe also:

1) Resets the GF flip-flop.
2) Resets the OUB flip-flop if no other instruction has been initiated.
3) Sets the Zero Indicator if the subtract resultant is zero.
4) Sets the Sign Indicator if bit zero of the subtract resultant is one.
5) Sets the Overflow Indicator if the range of AQ is exceeded.
6) Sets the Carry Indicator if a carry out of bit zero occurs, or resets it if no carry.
7) K_{72} adder control is reset.
8) Generates \$ LAST, which is sent to the CU to indicate completion of the SBAQ instruction.

The SBLAQ instruction is identical to this instruction with the exception that the Overflow Indicator is not affected for SBLAQ.

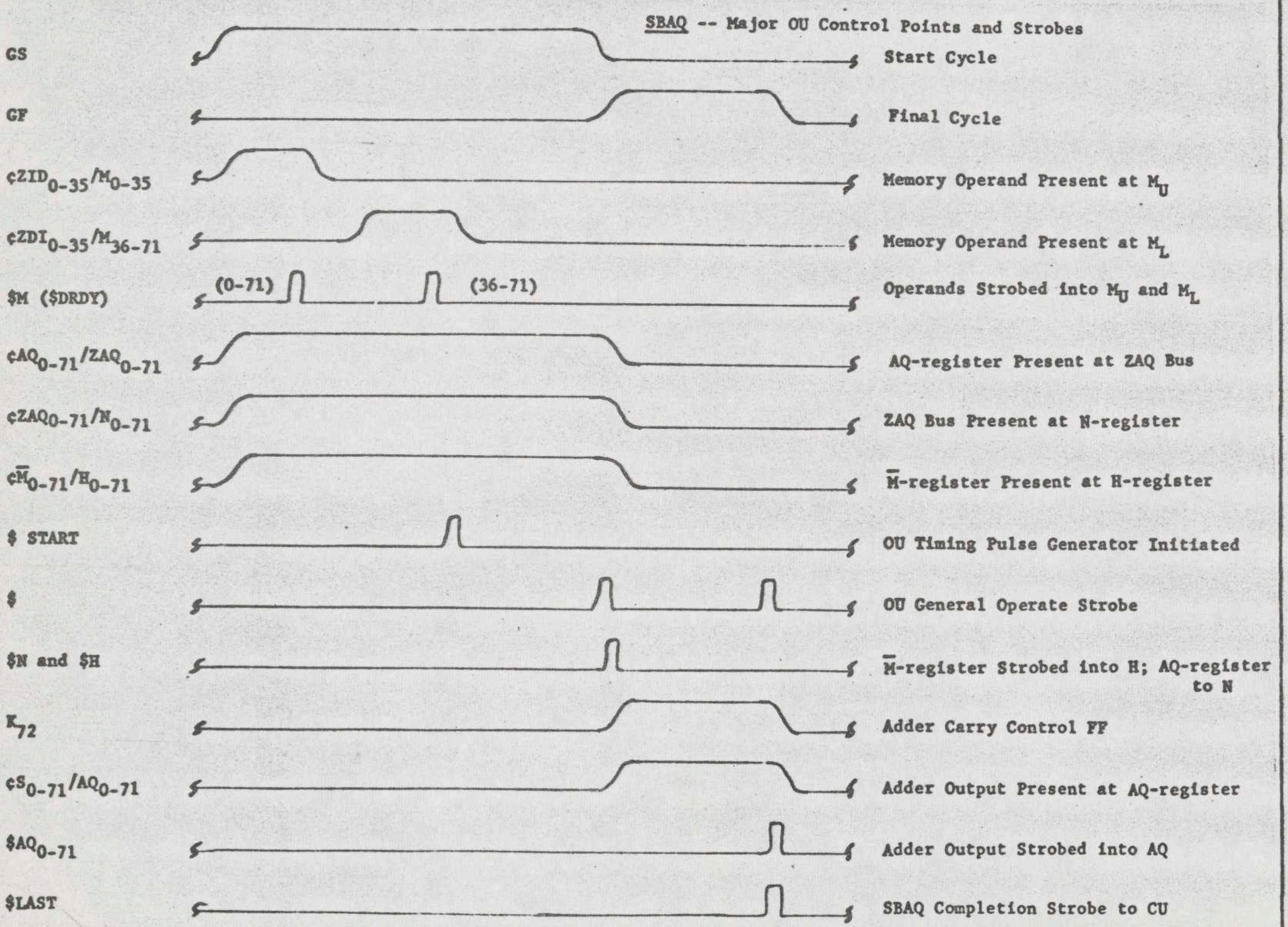

SBAQ -- Major OU Control Points and Strobes
GS
Start Cycle
GF
Final Cycle
¢ZID_{0-35}/M_{0-35}
Memory Operand Present at M_U
¢ZDI_{0-35}/M_{36-71}
Memory Operand Present at M_L
\$M (\$DRDY)
(0-71)
(36-71)
Operands Strobed into M_U and M_L
¢AQ_{0-71}/ZAQ_{0-71}
AQ-register Present at ZAQ Bus
¢ZAQ_{0-71}/N_{0-71}
ZAQ Bus Present at N-register
¢$\bar{M}_{0-71}/H_{0-71}$
$\bar{M}$-register Present at H-register
\$ START
OU Timing Pulse Generator Initiated
\$
OU General Operate Strobe
\$N and \$H
$\bar{M}$-register Strobed into H; AQ-register to N
K_{72}
Adder Carry Control FF
¢S_{0-71}/AQ_{0-71}
Adder Output Present at AQ-register
\$$AQ_{0-71}$
Adder Output Strobed into AQ
\$LAST
SBAQ Completion Strobe to CU

Fig. 12.3.1.2A

FIG. 12.3.1.2a SBAQ Instruction, Timing Diagram

12.3.2 FIXED POINT STORE GROUP

This group of 31 instructions constitute the majority of store instructions executed in the OU. The major data paths and control points for these instructions are shown in Fig. 12.3.2. In the execution of these instructions, the contents of some OU-register are transferred to the ZDO bus (data out bus) via the ZAQ switch or the S-adder, and the ZDO switch.

To illustrate the control used in the execution of these instructions, several instructions will be explained in detail. For the detailed control of all instructions in this group, refer to the OU Specification Charts.

In the explanation of all the example instructions, the following conditions will be assumed:

1) The instruction has been strobed into the O-register.

2) The GS and GC cycle flip-flops have been set.

3) The DDF flip-flop has been set indicating the operation decode has settled.

NOTE: Reference to Fig. 12.3.2 and the partial timing diagram at the end of each example will aid in the understanding of the instruction execution and implementation.

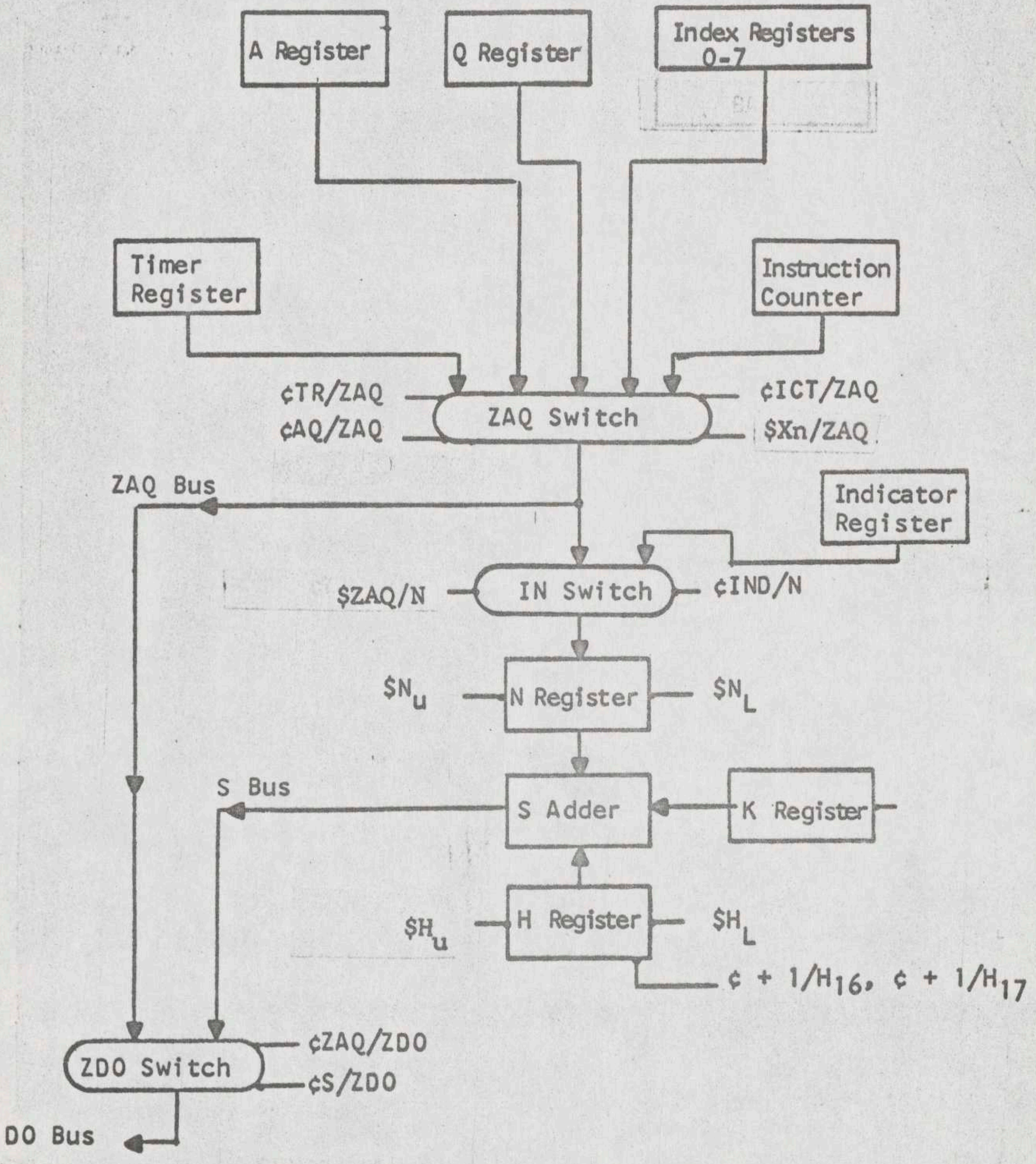

A Register
Q Register
Index Registers 0-7
Timer Register
Instruction Counter
¢TR/ZAQ
¢AQ/ZAQ
ZAQ Switch
¢ICT/ZAQ
$Xn/ZAQ
ZAQ Bus
Indicator Register
$ZAQ/N
IN Switch
¢IND/N
$N_u
N Register
$N_L
S Bus
S Adder
K Register
$H_u
H Register
$H_L
¢ + 1/H16, ¢ + 1/H17
ZDO Switch
¢ZAQ/ZDO
¢S/ZDO
DO Bus

Fig. 12.3.2 Major Data Paths & Control Points for the Fixed Point Store Group of Instructions (OU)

12.3.2.1 Single Precision

The instructions comprising the single precision portion of the Fixed-Point Store Group are:

STX_n, STA, STQ, STBA, STZ, STT, STCA, STCQ, STBQ,

SXL_n, SMIC, STC1, STI, STC2,

The last three of these instructions are different only in that they are stored via the S-adder, whereas the rest are stored via the ZAQ bus. One other difference is that $SDY for these three instructions is $ Reset GS delayed and for all other stores and RAR's $SDY is Set GS or Reset (GF).

Example 1: Store Q (STQ)

During this instruction the contents of the Q-register are stored in a memory location Y via the ZAQ bus.

The ZAQ and ZDO switches are enabled by the presence of the STQ decode together with the Byte Control flip-flop (BCFF) reset, the GC cycle flip-flop set, and the control point ¢YS/ZDO not enabled. The GC cycle will remain set from the beginning of the GS cycle to the end of the GF cycle for this store instruction. The ZAQ and ZDO switches are enabled by control points ¢AQ_{36-71}/ZAQ_{0-35} and ¢ZAQ_{0-35}/DO_{0-35}.

When the $ Set GS pulse was generated for this instruction, it was delayed by about 40 nanoseconds to become $YRY and becomes $SDY (store data ready). $YRY sets the operand ready flip-flop, and is returned to the CU to signal that a new operation code may be sent to the OU. YRY set allows the

12.3.2.1 - contd

$ START pulse to be generated and enter the Timing Pulse Generator to later produce the first operate strobe $. $ START sets the OU busy flip-flop, OUB. $SDY is sent to the CU to initiate the memory store cycle prior to the occurrence of the first $ strobe in the OU.

When the $ START pulse propogates the medium delay elements the first operate strobe $ occurs. This causes the following events:

1) The GS cycle to reset and the GF cycle to set.

2) The O-register Busy flip-flop (ORB) to reset.

3) The $ strobe re-enters the Timing Pulse Generator, but is inhibited from exiting the delay line network.

The operand will be stored in the Data Out Buffer, in the Interface Frame, by $INT. $INT will also generate a $DS and send it to the OU. $DS is "ORed" into the Timing Pulse Generator and exits on the general strobe ($) lines. This pulse:

1) Resets the GF and GC cycle flip-flops.

2) Resets the OUB flip-flop if no other instruction has been initiated.

3) Generates $LAST, which is sent to the CU to indicate completion of the STQ instruction.

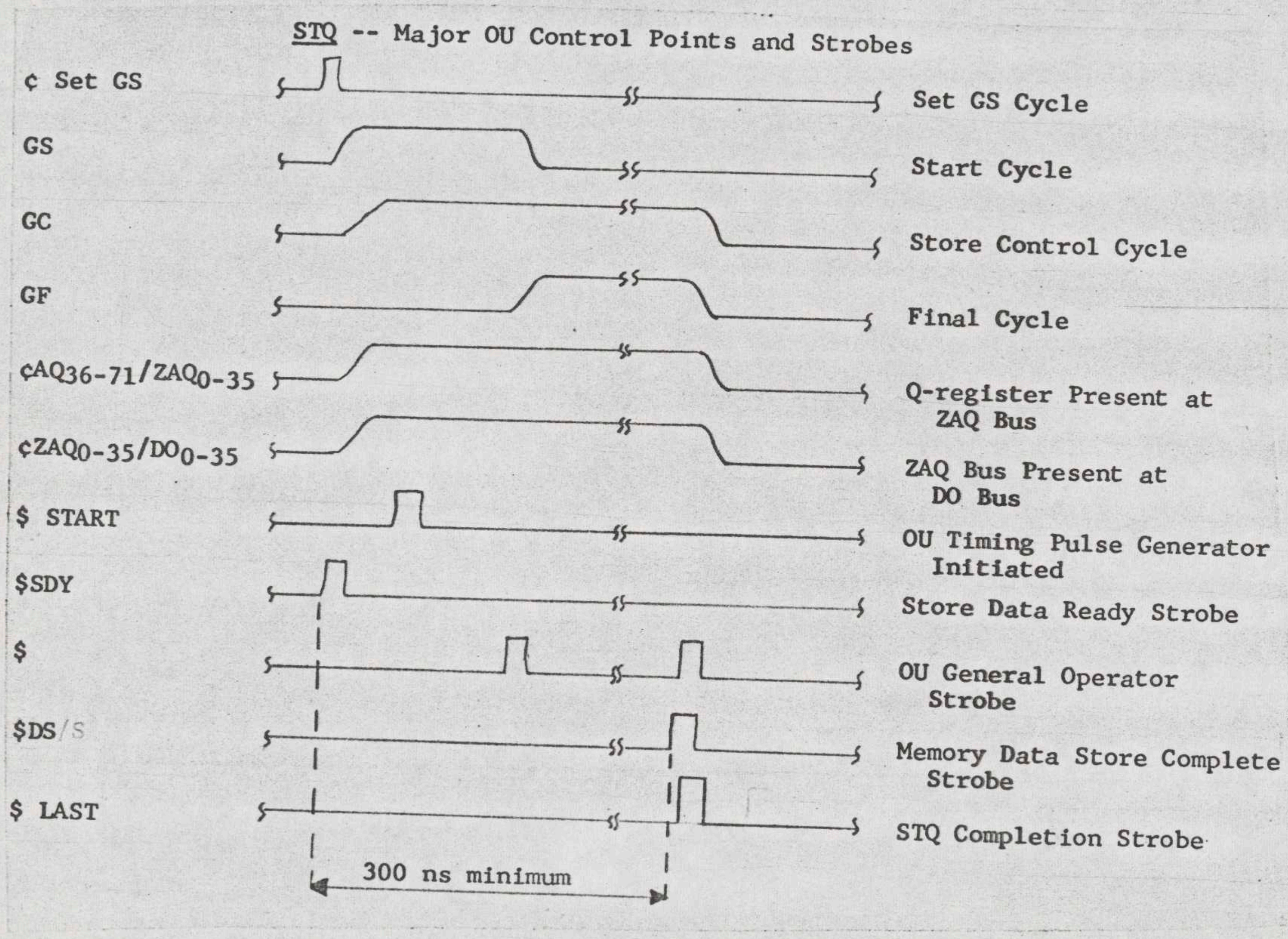

STQ -- Major OU Control Points and Strobes
¢ Set GS
GS
GC
GF
¢AQ36-71/ZAQ0-35
¢ZAQ0-35/DO0-35
$ START
$SDY
$
$DS/S
$ LAST
Set GS Cycle
Start Cycle
Store Control Cycle
Final Cycle
Q-register Present at ZAQ Bus
ZAQ Bus Present at DO Bus
OU Timing Pulse Generator Initiated
Store Data Ready Strobe
OU General Operator Strobe
Memory Data Store Complete Strobe
STQ Completion Strobe
300 ns minimum

Fig 12.3.2.1

FIG. 12.3.2.1 STQ Instruction, Timing Diagram

12.3.2.1 - contd

Example 2: Store Instruction Counter Plus 1 (STC1)

During this instruction the contents of the Instruction Counter C are incremented by one and stored via the adder in bit positions 0-17 of a memory location Y. The Indicator Register is stored into bit positions 18-28 of the memory location Y at the same time the (ICT_c+1) is stored.

When the STC1 decode is complete and the GS cycle has been set, the ZAQ and IN switches become enabled. ZAQ is enabled by ¢ICT_c/ZAQ_{0-17} and IN is enabled by ¢ZAQ_{0-35}/N_{0-35} and ¢IND/N_{18-35}. These enabled switches allow the Instruction Counter and the Indicator Register to be present at the N-register input. At the same time that the ZAQ and IN switches are enabled, ¢ + $1/H_{17}$ is enabled on the input gate to bit 17 of the H-register.

When the $ Set GS pulse was generated for this instruction, it was delayed by about 40 nanoseconds to become $YRY. $YRY sets the operand ready flip-flop YRY, and is returned to the CU. YRY set allows the $ START pulse to be generated and enter the OU Timing Pulse Generator to later produce the operate strobe $. $ START sets the OU busy flip-flop OUB. $ reset GS is delayed about 180 nanoseconds to become $SDY.

$SDY is sent to the CU to initiate the store cycle. The $SDY strobe for the three instructions which are stored via the adder output bus anticipates the fact the OU will have the proper data on the DO bus by the time the Interface Frame is ready to accept it.

12.3.2.1 -contd

GS

When the $ START pulse propogates the medium delay elements the first operate strobe $ occurs. This causes the following events:

1) The GS cycle to reset and the GF cycle to set.
2) $N strobes the Instruction Counter C into bit positions 0-17 of N, strobes the Indicator Register into bit positions 18-28 of N, and strobes zeros into the balance of the N-register.
3) $H strobes all zeros into the H-register with the exception of bit 17 which is set to a one.
4) The ORB flip-flop is reset.
5) The $ strobe re-enters the Timing Pulse Generator, but since a store is being executed it will be inhibited from exiting onto the general $ lines.

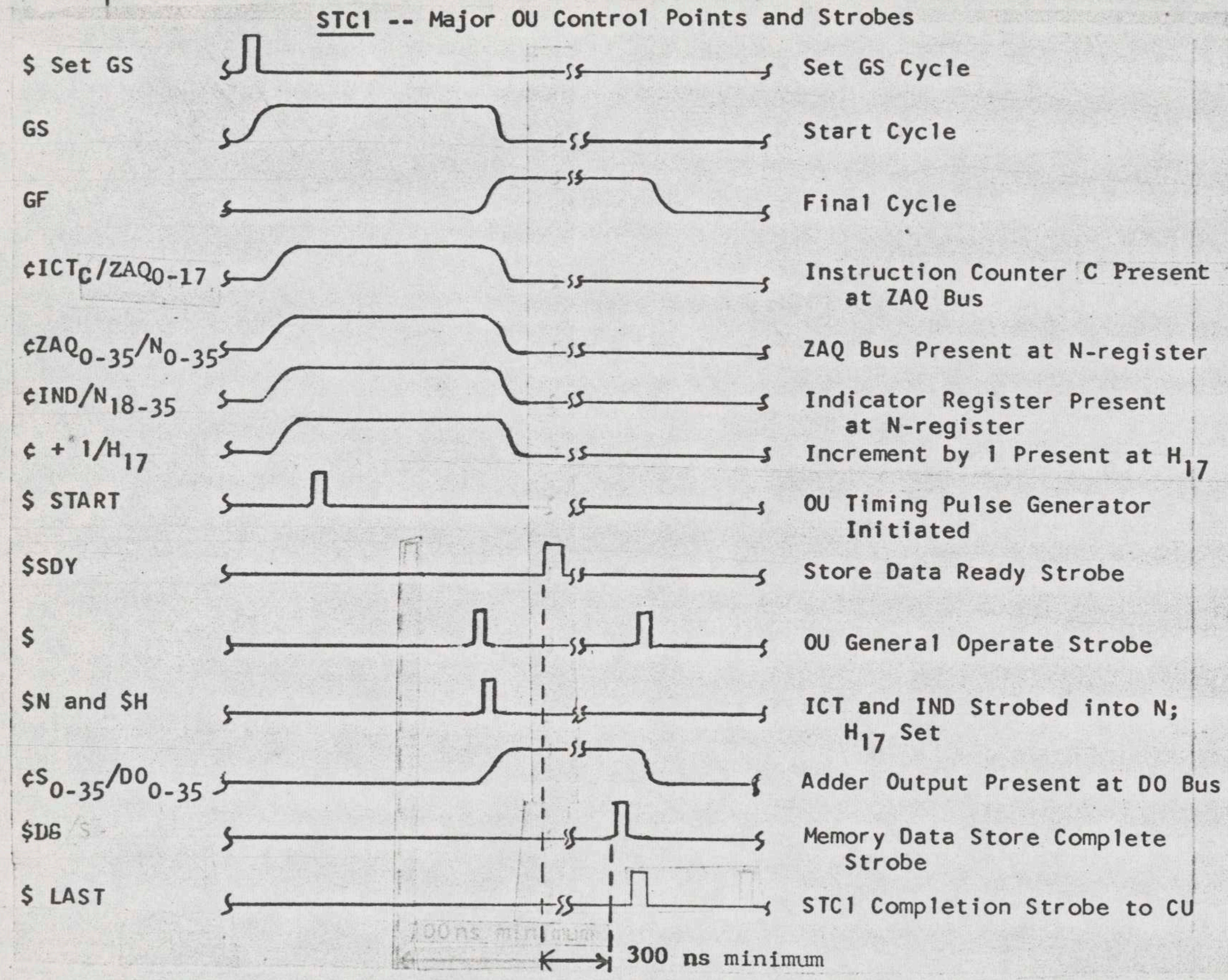
GF

When the GF sets, the ZDO switch is enabled by ¢S_{0-35}/DO_{0-35}. This allows the adder output (which is the ICT_c+1 and the Indicator Register) to be present on the DO bus.

The operand will be stored in the Data Out Buffer, in the Interface Frame, by $INT. $INT will also generate a $DS and send it to the OU. $DS is "ORed" into the Timing Pulse Generator and exits on the general strobe ($) lines. This pulse:

1) Resets the GF cycle.
2) Resets the OUB flip-flop if no other instruction has been initiated.
3) Generates $ LAST, which is sent to the CU to indicate completion of STC1 instruction.

The STC2 instruction is similar to the STC1 except the ICT_c is incremented by 2 and the Indicator Register is not stored with (ICT+2).



STC1 -- Major OU Control Points and Strobes

$ Set GS
Set GS Cycle
GS
Start Cycle
GF
Final Cycle
¢ICTC/ZAQ0-17
Instruction Counter C Present at ZAQ Bus
¢ZAQ0-35/N0-35
ZAQ Bus Present at N-register
¢IND/N18-35
Indicator Register Present at N-register
¢ + 1/H17
Increment by 1 Present at H17
$ START
OU Timing Pulse Generator Initiated
$SDY
Store Data Ready Strobe
$
OU General Operate Strobe
$N and $H
ICT and IND Strobed into N; H17 Set
¢S0-35/DO0-35
Adder Output Present at DO Bus
$DG
Memory Data Store Complete Strobe
$ LAST
STC1 Completion Strobe to CU
300 ns minimum

Fig 12.3.2.1a

FIG. 12.3.2.1a STC1 Instruction Timing Diagram

12.3.2.2 Double Precision

The instructions comprising the double precision portion of the Fixed-point Store Group are SMCM, LACL, and STAQ. In these instructions the contents of AQ are stored via the ZAQ bus.

Example 1: Set Memory Controller Mask Register (SMCM)

During this instruction the contents of the AQ are stored in a Memory Controller Interrupt Mask Register and a Memory Controller Access Mask Register as specified by bits 0-2 of the effective address Y.

When the SMCM decode has settled and the GS cycle is set, the GC cycle flip-flop sets. This allows the ZAQ and ZDO switches to be enabled by $¢AQ_{0-35}/ZAQ_{0-35}$ and $¢ZAQ_{0-35}/DO_{0-35}$. These switches enable the A-register to be present at the DO bus.

When the $ Set GS pulse was generated for this instruction, it was delayed by about 40 nanoseconds to become $YRY and it becomes $SDY. $YRY sets the operand ready flip-flop YRY, and is returned to the CU. YRY set allows the $ START pulse to be generated and enter the OU Timing Pulse Generator to later produce the operate strobe $. $ START sets the OU busy flip-flop OUB.

$SDY is sent to the CU to initiate the store cycle prior to the occurrence of the first $ strobe in the OU.

GS

When the $ START pulse propogates the medium delay elements the operate strobe $ occurs. This causes the following events:

1) The GS cycle to reset and the GT cycle to set.

12.3.2.2 - contd

2) The O-register busy flip-flop ORB, to reset.

3) A level consisting of DP stores and GT goes high.

4) The $ strobe re-enters the Timing Pulse Generator, but since a store is being executed it is inhibited from exiting onto the general $ lines.

GT

After a minimum time of 300 nanoseconds after $SDY, the first half of the store operation will be complete and the first $DS will be received from the Interface Frame to signal this fact. $DS will produce a $ strobe which will:

1) Reset the GC and GT cycle flip-flops.

2) Set the GF and GFT cycle flip-flops.

3) Cause the DP stores and GT level to go low after about 60 nanoseconds but only after $ strobe has "ANDed" with it to produce $DP which is delayed by approximately 270 nanoseconds before being sent to the Interface Frame to signify that the second data word is ready to be stored.

GF

When GF set, the ZAQ and ZDO switches became enabled by control points $\text{¢AQ}_{36-71}/\text{ZAQ}_{0-35}$ and $\text{¢ZAQ}_{0-35}/\text{DO}_{0-35}$. This allows the Q-register outputs to be present at the DO bus before the OU sends the $DP pulse to the Interface Frame for the second half store.

12.3.2.2 - contd

The $DP strobes the data into the Data Out Buffer bits 36-71 and is returned as the second $DS indicating the second half data has been received. This $DS will generate a $ strobe pulse which will:

1) Rest the GF and GFT cycles.

2) Reset the OUB flip-flop if no other instruction has been initiated.

3) Generates $ LAST, which is sent to the CU to indicate completion of the SMCM instruction.

SMCM -- Major OU Control Points and Strobes

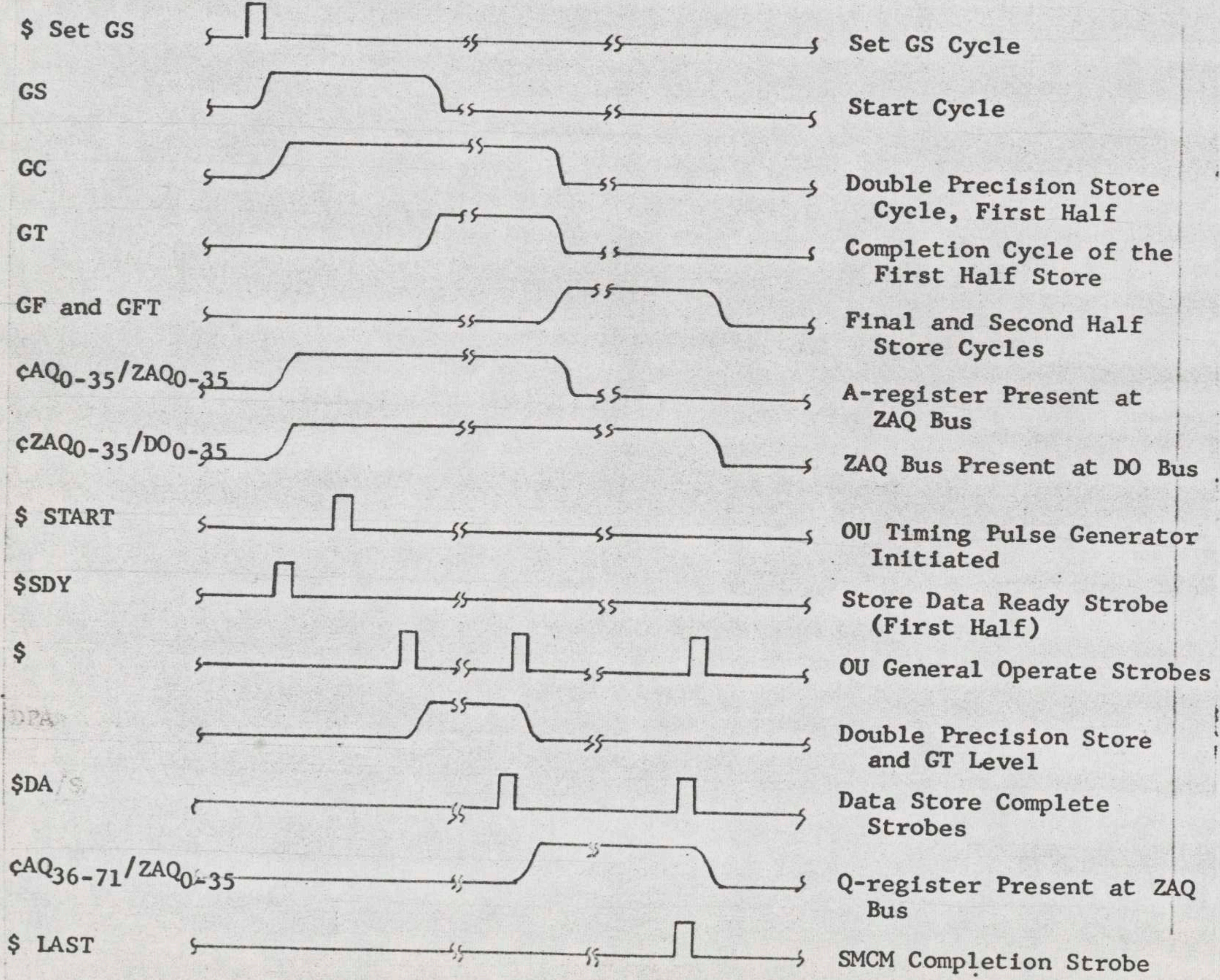

$ Set GS
Set GS Cycle
GS
Start Cycle
GC
Double Precision Store Cycle, First Half
GT
Completion Cycle of the First Half Store
GF and GFT
Final and Second Half Store Cycles
¢AQ0-35/ZAQ0-35
A-register Present at ZAQ Bus
¢ZAQ0-35/DO0-35
ZAQ Bus Present at DO Bus
$ START
OU Timing Pulse Generator Initiated
$SDY
Store Data Ready Strobe (First Half)
$
OU General Operate Strobes
DPA
Double Precision Store and GT Level
$DA/S
Data Store Complete Strobes
¢AQ36-71/ZAQ0-35
Q-register Present at ZAQ Bus
$ LAST
SMCM Completion Strobe

FIG. 12.3.2.2 SMCM Instruction, Timing Diagram

12.3.3 READ-ALTER-REWRITE GROUP

This group of 52 instructions may be considered to be a composite of the single precision portion of the Add Group and Store Group instructions previously presented. They look like the Add Group in that a memory word is pulled into the OU and it is altered in the S-adder with the contents of some register (A, Q, or X_n). The storing of the adder results to the original memory location then make these instructions look similar to the Store Group. The instructions comprising the RAR Group are:

ASQ, ASA, ASX_n, SSQ, AOS, SSA, SSX_n, ANSQ, ANSA,

$ANSX_n$, ORSQ, ORSA, $ORSX_n$, ERSQ, ERSA, $ERSX_n$, STAC.

These instructions are all single precision, all use the S-adder, and are all executed with a GS-GF cycle sequence. The major data paths and control points for these instructions are shown in Fig. 12.3.3.

To illustrate the control used in the execution of these instructions, several instructions will be explained in detail. For the detailed control of all instructions in this group, refer to the OU Specification Charts 43D140232, pages 2, 3, 5, and 11.

In the explanation of all the example instructions, the following conditions will be assumed:

1) The instruction has been strobed into the O-register.
2) The GS cycle flip-flop has been set.
3) The DDF flip-flop has been set indicating the operation decode has settled.

NOTE: Reference to Fig. 12.3.3 and the partial timing diagram at the end of each example will aid in the understanding of the instruction execution and implementation.

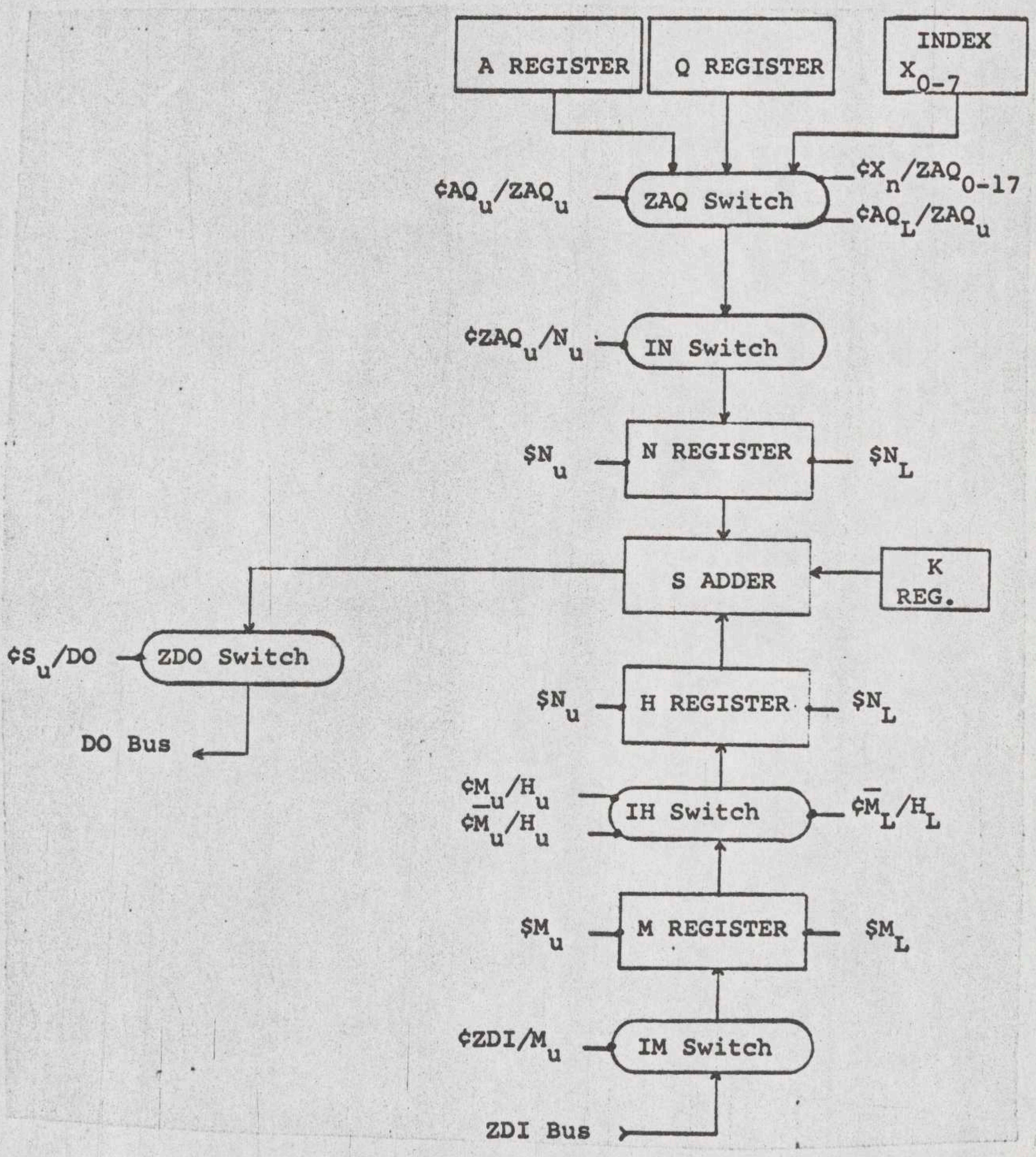

A REGISTER
Q REGISTER
INDEX X_{0-7}
¢AQ_u/ZAQ_u
ZAQ Switch
¢X_n/ZAQ_{0-17}
¢AQ_L/ZAQ_u
¢ZAQ_u/N_u
IN Switch
\$$N_u$
N REGISTER
\$$N_L$
S ADDER
K REG.
¢S_u/DO
ZDO Switch
DO Bus
\$$N_u$
H REGISTER
\$$N_L$
¢M_u/H_u
¢$\overline{M}_u$/H_u
IH Switch
¢$\overline{M}_L$/H_L
\$$M_u$
M REGISTER
\$$M_L$
¢ZDI/M_u
IM Switch
ZDI Bus

Fig. 12.3.3 Major Data Paths and Control Points for the Read-Alter-Rewrite Group of Instructions (OU)

12.3.3 - contd

Example 1: Add One to Storage (AOS)

During this instruction the contents of a memory location Y are incremented by one and the results are placed back in Y.

The IM switch is enabled by the presence of the AOS decode together with DPFF, BCFF, and DIC all reset. The control points ¢ZDI_{0-17}/M_{0-17}, ¢ZDI_{18-27}/M_{18-27}, and ¢ZDI_{28-35}/M_{28-35} are enabled to allow the ZDI bus to be present at the M-register inputs. About the same time that IM is enabled, the IH_switch is enabled with ¢M_{0-35}/H_{0-35} and ¢M_{36-71}/H_{36-37}. This will allow the memory operand to be present at bits 0-35 of the H-register and all one's at bit positions 36-71. All one's are forced into the lower half of H so that when an add is performed with an initial carry, the carry will ripple down to bit 35 permitting +1 to be added to the memory operand.

$ set GS generates $SDY and sends it to the CU as a request for the AOS operand to be retrieved from memory. After about 1100 nanoseconds the operand ready strobe $DRDY is received from the CU.

The reception of $DRDY and the double precision flip-flop not set allows the M-register strobe \$$M_{0-71}$ to be generated and strobe the memory operand into M. $ START is generated in the normal manner.

GS

When the $ START pulse propogates the medium delay elements, the operate strobe $ occurs. This causes the following events:

12.3.3 - contd

1) The GS cycle to reset and the GF cycle to set.
2) $H strobes the memory operand into H (0-35). All one's are strobed into H (36-71).
3) $N is generated, but since the IN switch is not enabled all zeros are loaded into N.
4) The adder carry control flip-flop K_{72} is set.
5) The Zero and Sign Indicators are reset.
6) The (RAR·GS·$\overline{\text{GF}}$) level gates $ to produce the $DP strobe. This pulse delayed 350 nanoseconds will be used for the storage of the altered AOS operand in the Data Out Buffer.

GF

When GF set, the control point ¢S_{0-35}/DO_{0-35} became enabled. This permits the adder outputs to be present at the DO bus. The $DP will strobe the altered word into the Data Out Buffer and will be returned as $DS. This $DS will produce a $ strobe which will:

1) Reset the GF cycle.
2) Reset the OUB flip-flop if no other instruction has been initiated.
3) Set the Zero Indicator if the adder output is zero.

12.3.3 - contd

4) Set the Sign Indicator if bit zero of the adder is a one.

5) Set the Carry Indicator if a carry was generated out of bit zero, or reset it if no carry.

6) Set the Overflow Indicator if the range of Y is exceeded.

7) Reset the K_{72} carry flip-flop.

8) Generate $ LAST, which is sent to the CU to signal completion of the AOS instruction.

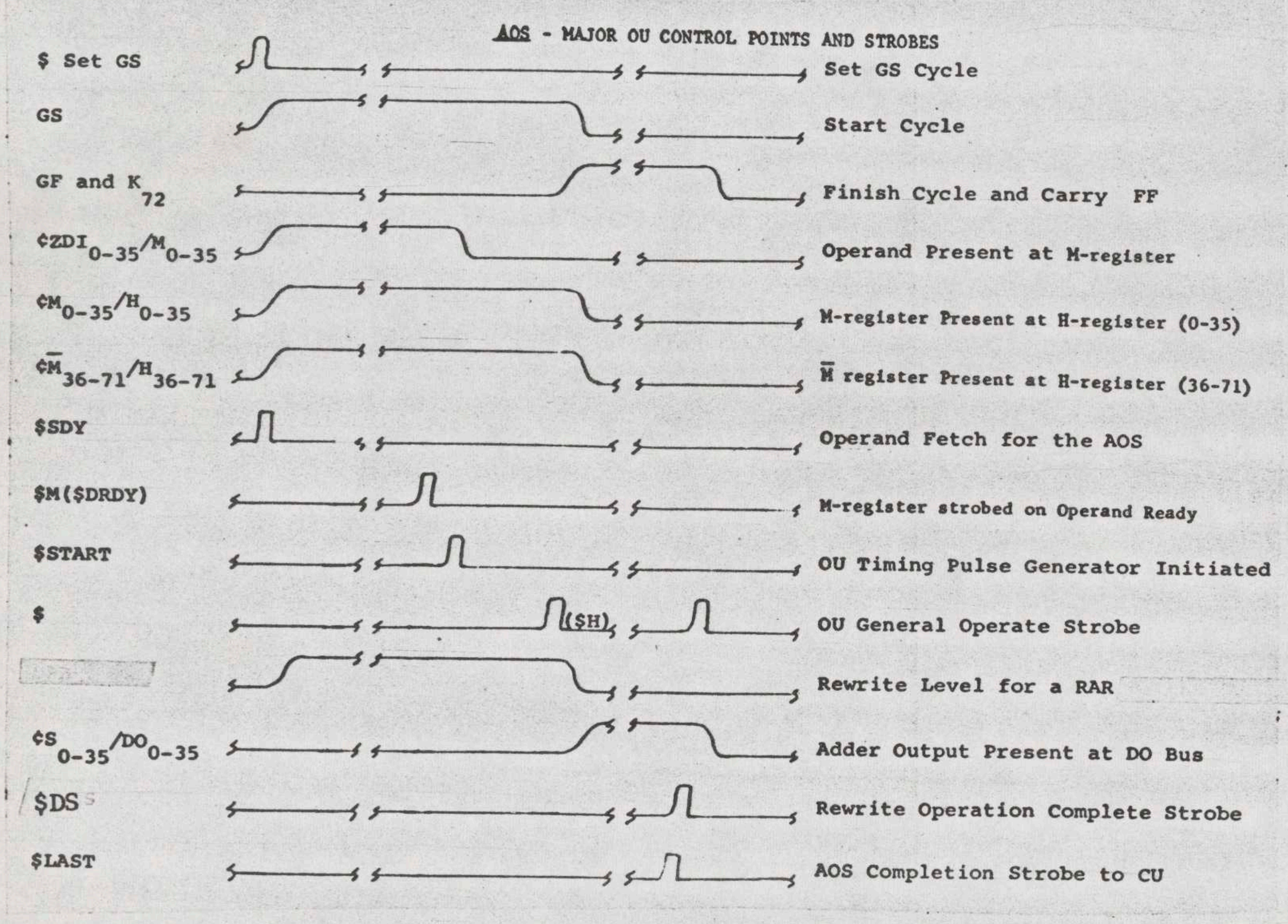

AOS - MAJOR OU CONTROL POINTS AND STROBES
$ Set GS
Set GS Cycle
GS
Start Cycle
GF and K$_{72}$
Finish Cycle and Carry FF
¢ZDI$_{0-35}$/M$_{0-35}$
Operand Present at M-register
¢M$_{0-35}$/H$_{0-35}$
M-register Present at H-register (0-35)
¢$\overline{M}_{36-71}$/H$_{36-71}$
$\overline{M}$ register Present at H-register (36-71)
$SDY
Operand Fetch for the AOS
$M($DRDY)
M-register strobed on Operand Ready
$START
OU Timing Pulse Generator Initiated
$
($H)
OU General Operate Strobe
Rewrite Level for a RAR
¢S$_{0-35}$/DO$_{0-35}$
Adder Output Present at DO Bus
$DS
Rewrite Operation Complete Strobe
$LAST
AOS Completion Strobe to CU

FIG. 12.3.3a AOS Instruction, Timing Diagram

12.3.3 - contd

Example 2: Exclusive "OR" to Storage Q (ERSQ)

During this instruction the exclusive "OR" function is performed using the contents of a memory location Y and the contents of the Q-register. The results are stored in location Y.

The IM switch is enabled by the presence of the ERSQ decode together with DPFF, BCFF, and DIC all reset. Control points ¢ZDI$_{0-17}$/M$_{0-17}$, ¢ZDI$_{18-27}$/M$_{18-27}$, and ¢ZDI$_{28-35}$/M$_{28-35}$ are enabled to allow the ZDI bus to be present at the M-register inputs. About the same time the IM switch is enabled the IH, ZAQ, and IN switchs are enabled. IH is enabled by ¢M$_{0-35}$/H$_{0-35}$; ZAQ by ¢AQ$_{36-71}$/ZAQ$_{0-35}$; and IN by ¢ZAQ$_{0-35}$/N$_{0-35}$. These enabled switches will allow the operands to be present at the H- and N-register inputs.

$ set GS generates a $SDY which is sent to the CU as a request for the ERSQ operand to be retrieved from memory. After about 1100 nanoseconds the operand ready strobe $DRDY is received from the CU.

The reception of $DRDY and the double precision flip-flop not set allows the M-register strobe $\$M_{0-71}$ to be generated and strobe the memory operand into M. $ START is generated in the normal manner.

GS

When the $ START pulse propogates the medium delay elements, the operate strobe $ occurs. This causes the following events:

12.3.3 - contd

1) The GS cycle to reset and the GF cycle to set.
2) $H strobes the memory operand into H.
3) $N strobes the Q-register operand into N.
4) The adder control flip-flop K_1 is set to force an exclusive "OR" to be performed.
5) The Zero and Sign indicators are reset.
6) The (RAR·GS·$\overline{GF}$) level which is high gates $ to produce the $DP strobe. This pulse delayed 350 nanoseconds will be used for the rewrite of the altered ERSQ operand.

GF

When GF set, the control point ¢S$_{0-35}$/DO$_{0-35}$ became enabled. This will allow the adder outputs to be present at the DO bus. The $DP will strobe the altered word into the Data Out Buffer and will be returned as $DS. This $DS will produce a $ strobe which will:

1) Reset the GF cycle.
2) Reset the OUB flip-flop if no other instruction has been initiated.
3) Set the Zero Indicator if the adder output is zero.
4) Set the Sign Indicator if bit zero of the adder is a one.
5) Reset K_1 of the adder control.
6) Generates $ LAST, which is sent to the CU to signal completion of the ERSQ instruction.

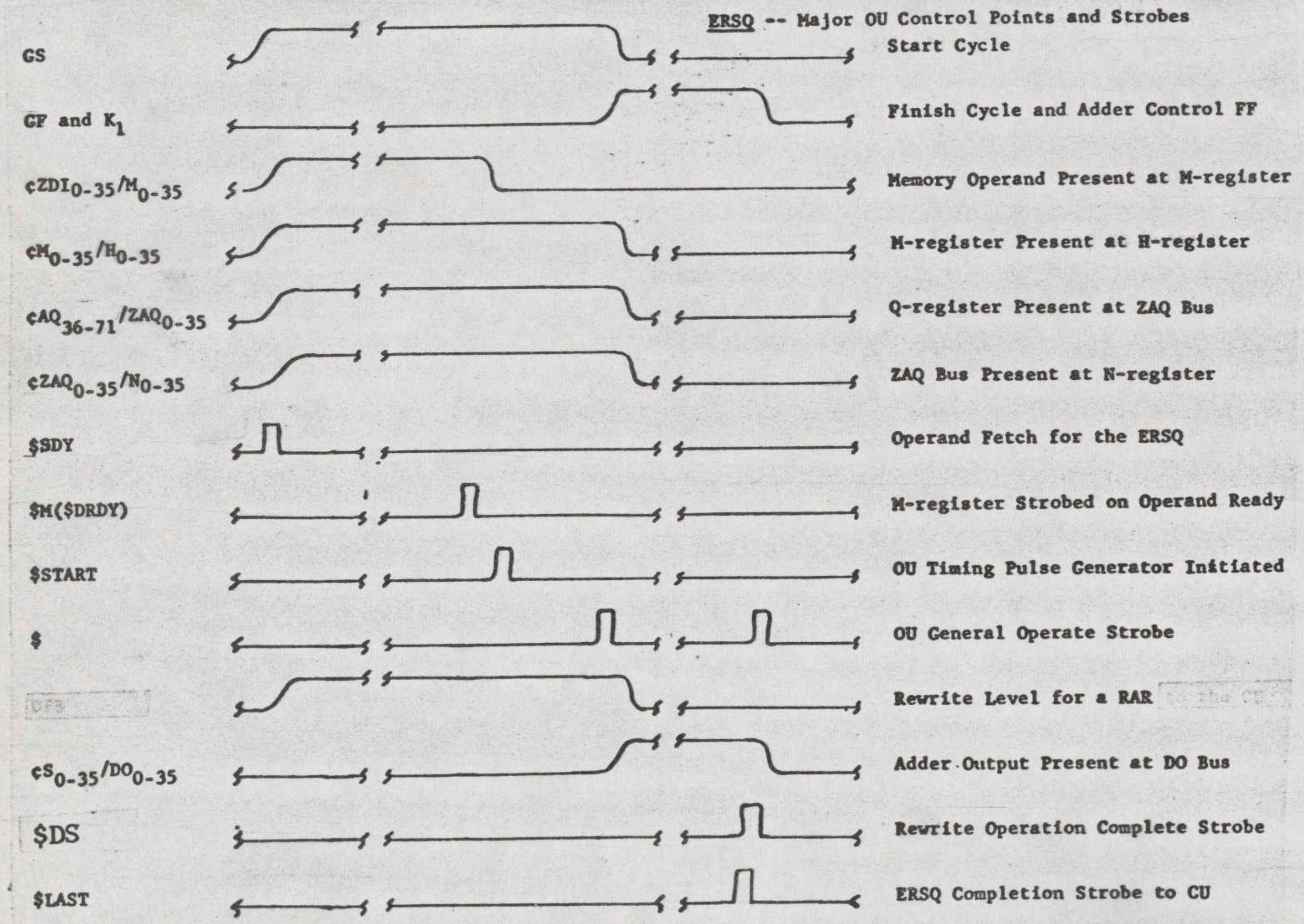

ERSQ -- Major OU Control Points and Strobes
GS
Start Cycle
GF and K_1
Finish Cycle and Adder Control FF
$¢ZDI_{0-35}/M_{0-35}$
Memory Operand Present at M-register
$¢M_{0-35}/H_{0-35}$
M-register Present at H-register
$¢AQ_{36-71}/ZAQ_{0-35}$
Q-register Present at ZAQ Bus
$¢ZAQ_{0-35}/N_{0-35}$
ZAQ Bus Present at N-register
$SDY
Operand Fetch for the ERSQ
$M($DRDY)
M-register Strobed on Operand Ready
$START
OU Timing Pulse Generator Initiated
$
OU General Operate Strobe
OFB
Rewrite Level for a RAR to the CU
$¢S_{0-35}/DO_{0-35}$
Adder Output Present at DO Bus
$DS
Rewrite Operation Complete Strobe
$LAST
ERSQ Completion Strobe to CU

Fig 12.3.3b

FIG. 12.3.3b EROS Instruction, Timing Diagram

12.3.4 SHIFT GROUP

This group of 12 instructions constitute the controllable shifting capabilities of this machine. These instructions permit left shifts, right shifts, and left rotations of the A-, Q-, or combined AQ-registers with from 0 to 127 bit displacements. Even though shifts of greater than 35 for single precision and shifts of greater than 71 for double precision are generally meaningless, they are allowed by the hardware.

All shifts are performed under control of the OU G-counter* which contains the number of shifts to be executed. The G-counter is loaded with bits 11-17 of the shift operand which specifies the number of shifts. The shift logic is implemented so that shifts of 16, 4, or 1 in that order may be performed during one shift or GO cycle. For each bit shifted, the shift count in G is decremented by one. This continues until the G count is zero. During this shift process there may be a right or left overshift of one bit position. In this event a Reverse flip-flop (REV) is set and the overshifted bit is retained and inserted back into the data word during the GN cycle used for overshift correction.

All shifts are performed by transferring the A-, Q-, or AQ-registers into the N-register and then switching the N-register outputs back into the appropriate inputs in order to accomplish the actual shift. When the shift operation is complete, the results are stored back in the A-, Q-, or AQ-register.

---

*Refer to paragraph 12.4 for a detailed description of the G-counter.

12.3.4 - contd

To illustrate the control used in the execution of these instructions, two instructions will be explained in detail. For the detailed control of all instructions in this group, refer to the OU Specification Chart.

In the explanation of the example instructions, the following conditions will be assumed.

1) The instruction has been strobed into the O-register.

2) The GS cycle flip-flop has been set.

3) The DDF flip-flop has been set indicating the operation decode has settled.

NOTE: Reference to Fig. 12.3.4 and the partial timing diagram at the end of each example will aid in the understanding of the instruction execution and implementation.

Major Data Paths and Control Points for the Shift Group of Instructions (OU)

C*-N to N Shift Control

1. ¢N_{-1}/N_0
2. ¢N_0/N_0
3. ¢N_0/N_{0-3}
4. ¢N_0/N_{0-15}
5. ¢N_{0-34}/N_{1-35}
6. ¢N_{0-31}/N_{4-35}
7. ¢N_{0-19}/N_{16-35}
8. ¢N_{35-70}/N_{36-71}
9. ¢N_{32-67}/N_{36-71}
10. ¢N_{20-55}/N_{36-71}
11. ¢N_0/N_{35}
12. ¢N_{0-3}/N_{32-35}
13. ¢N_{0-15}/N_{20-35}
14. ¢N_{1-35}/N_{0-34}
15. ¢N_{4-35}/N_{0-31}
16. ¢N_{16-35}/N_{0-19}
17. ¢$(N_{36}+N_{36A}/N_{35})$
18. ¢N_{36-39}/N_{32-35}
19. ¢N_{36-51}/N_{20-35}
20. ¢N_0/N_{71}
21. ¢N_{0-3}/N_{68-71}
22. ¢N_{0-15}/N_{56-71}
23. ¢N_{37-71}/N_{36-70}
24. ¢N_{40-71}/N_{36-67}
25. ¢N_{52-71}/N_{36-55}
26. ¢N_{72}/N_{71}
27. ¢N_{35}/N_0
28. ¢N_{71}/N_0
29. ¢$N_0 \oplus N_{35}/N_{35}$ (GTB)

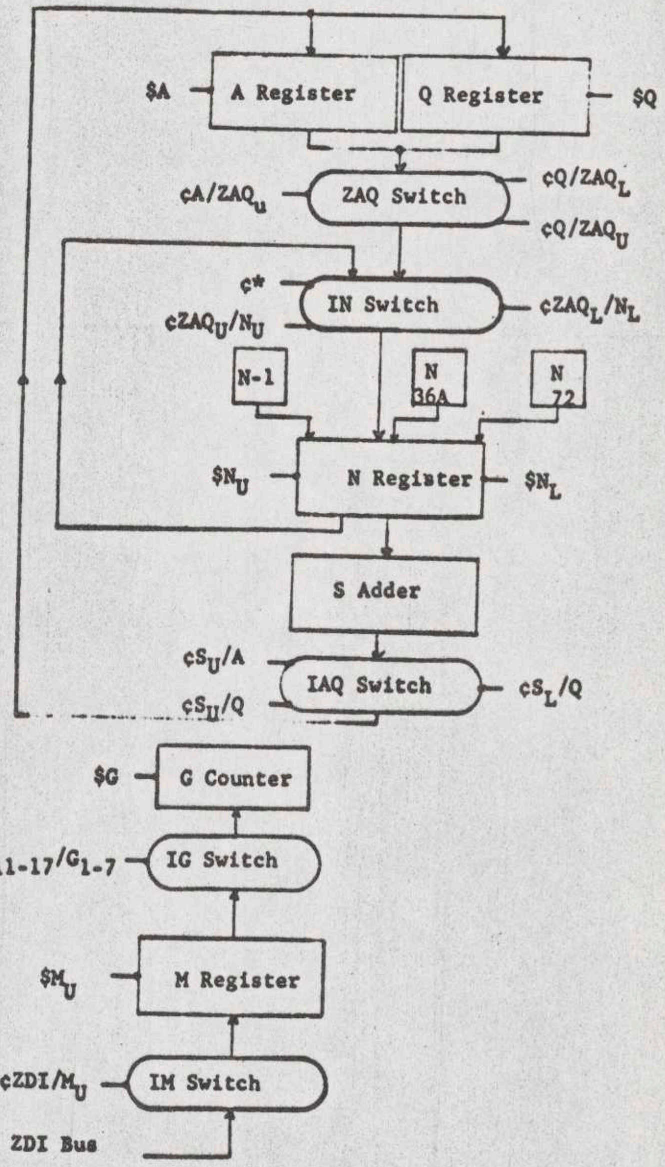

$A
A Register
Q Register
$Q
¢A/ZAQ_U
ZAQ Switch
¢Q/ZAQ_L
¢Q/ZAQ_U
¢*
¢ZAQ_U/N_U
IN Switch
¢ZAQ_L/N_L
N-1
N 36A
N 72
$$N_U$
N Register
$$N_L$
S Adder
¢S_U/A
¢S_U/Q
IAQ Switch
¢S_L/Q
$G
G Counter
¢M_{11-17}/G_{1-7}
IG Switch
$$M_U$
M Register
¢ZDI/M_U
IM Switch
ZDI Bus

Fig. 12.3.4

12.3.4 - contd

Example 1: A-register Left Shift (ALS)

During this instruction the contents of the A-register will be shifted left the number of positions specified by bits 11-17 of the ALS instruction word. The vacated positions will be filled with zeros and the final result will be placed in A. For this example a shift of 7 will be executed.

The IM switch is enabled by the presence of the ALS decode together with the DPFF, BCFF, and DIC all reset. The control points ¢ZDI$_{0-17}$/M$_{0-17}$, ¢ZDI$_{18-27}$/M$_{18-27}$, and ¢ZDI$_{28-35}$/M$_{28-35}$ are enabled to allow the ZDI bus to be present at the M-register inputs. Shortly after the IM switch is enabled the IG, ZAQ, and IN switches are enabled. IG is enabled by ¢M$_{11-17}$/G$_{1-7}$; ZAQ by ¢AQ$_{0-35}$/ZAQ$_{0-35}$; and IN by ¢ZAQ$_{0-35}$/N$_{0-35}$. These enabled switches allow the operands to be present at the N and G inputs.

When $DRDY is received from the CU, the double precision flip-flop not being set allows the M-register strobe \$M$_{0-71}$ to be generated and strobe the shift count into M. $ START is generated in the normal manner.

GS

When the $ START pulse propogates the medium delay elements the operate strobe $ occurs. This causes the following events:

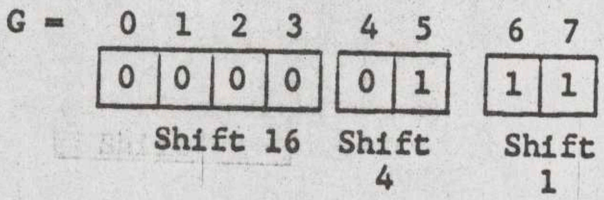
1) The GS cycle to reset and the GO cycle to set.

2) $N strobes the contents of A into N.

3) $H strobes all zeros into H.

12.3.4 - contd

4) \$G strobes the contents of M_{11-17} into G_{1-7} as shown:



5) The Zero, Sign, and Carry Indicators are all reset.

6) The \$ strobe re-enters the Timing Pulse Generator delay line network to produce another \$ strobe in approximately 200 nanoseconds.

(GO) (G = 7)

During this 200-nanosecond interval the first fast GO cycle is true and the control points are set up to execute the first shift. The G contents call for one $(01)_2$ shift of 4 and three $(11)_2$ shifts of 1. The greater shifts are executed first, so the shift of 4 will occur during this first GO cycle. The control point enabled for this is $¢N_{4-35}/N_{0-31}$. This control will allow bits 4-35 of N to be switched into bit positions 0-31 of N on the next \$N. Since the inputs to N_{32-35} will not be true at this time, zeros will be forced into the lower 4-bit positions of N.

When the \$ strobe does occur at the end of this first GO cycle, the following events occur:

12.3.4 - contd

1) \$N strobes the N-register inputs and performs the shift of 4.

2) \$G decrements the G-counter by 4.

3) The Carry Indicator will set if all of the 4-bits shifted out of N were not alike.

4) The GO cycle will be strobed but will remain set because the G count indicates another GO cycle is necessary.

5) The \$ pulse will re-enter the Timing Pulse Generator delay line network to produce another \$ in about 200 nanoseconds.

(GO) (G = 3)

During this 200 nanosecond interval the second GO cycle is true and the control points are set up to execute the second shift. At this time the G-counter still has bits 6 and 7 true which call for three shifts of 1. The shift control logic is set up in such a manner that a shift of 3 will be done by a shift of 4 and a reverse shift of 1. This speeds up the over-all shift by one cycle. The control point again enabled is ¢N_{4-35}/N_{0-31} which is the same control used for the first shift of 4. The set reverse control line comes high by the decode of (G count = 3) (GO) (ALS) and sets up the necessary condition for terminating the GO cycle and entering the GN cycle.

When the \$ strobe occurs at the end of the second GO cycle, the following events occur:

12.3.4 - contd

1) $N strobes the N-register inputs and performs the shift of 4.

2) $G decrements the G-counter to zero by resetting G_6 and G_7.

3) The Carry Indicator will set if any of the 4-bits shifted out of N are not all the same (zeros or ones) and it was not set during the first GO cycle.

4) The N_{-1} flip-flop is set or reset depending on whether N_3 was a one or a zero. This will later control the sign of N when the overshift correction is done and N_{-1} is restored to the N_0 position.

5) The GO cycle is reset and the GN cycle set.

6) The $ strobe will re-enter the Timing Pulse Generator to produce another $ in about 200 nanoseconds.

7) The reverse flip-flop (REV) is set.

GN

During the GN cycle the control points are set up to execute the overshift correction. For this case the correction is a right shift of one and the insertion of the overshifted bit contained in N_{-1} back into N_0. The control points enabled are ¢N_{0-31}/N_{1-35} and ¢N_{-1}/N_0.

When the $ strobe occurs at the end of the GN cycle, the following events occur:

12.3.4 - contd

1) $N strobes the N-register inputs and performs a right shift of one and pulls in bit zero.

2) The Reverse flip-flop (REV) is reset.

3) The GN cycle is reset and the GF cycle is set.

4) The $ strobe is re-enter the Timing Pulse generator to produce another $ in about 400 nanoseconds.

GF

At this point the shift is complete and all that remains is to restore the shifted operand back to A-register. This restore is done via the S-adder. The IAQ switch control enabled is $¢S_{0-35}/AQ_{0-35}$. When the $ strobe occurs at the end of GF, the following events occur:

1) $\$AQ_{0-35}$ restores the shifted A operand.

2) The Zero Indicator is set if the contents of A = 0.

3) The Sign Indicator is set if bit zero of A = 1.

4) The GF cycle is reset.

5) It generates $ LAST, which is sent to the CU to signal completion of the ALS instruction.

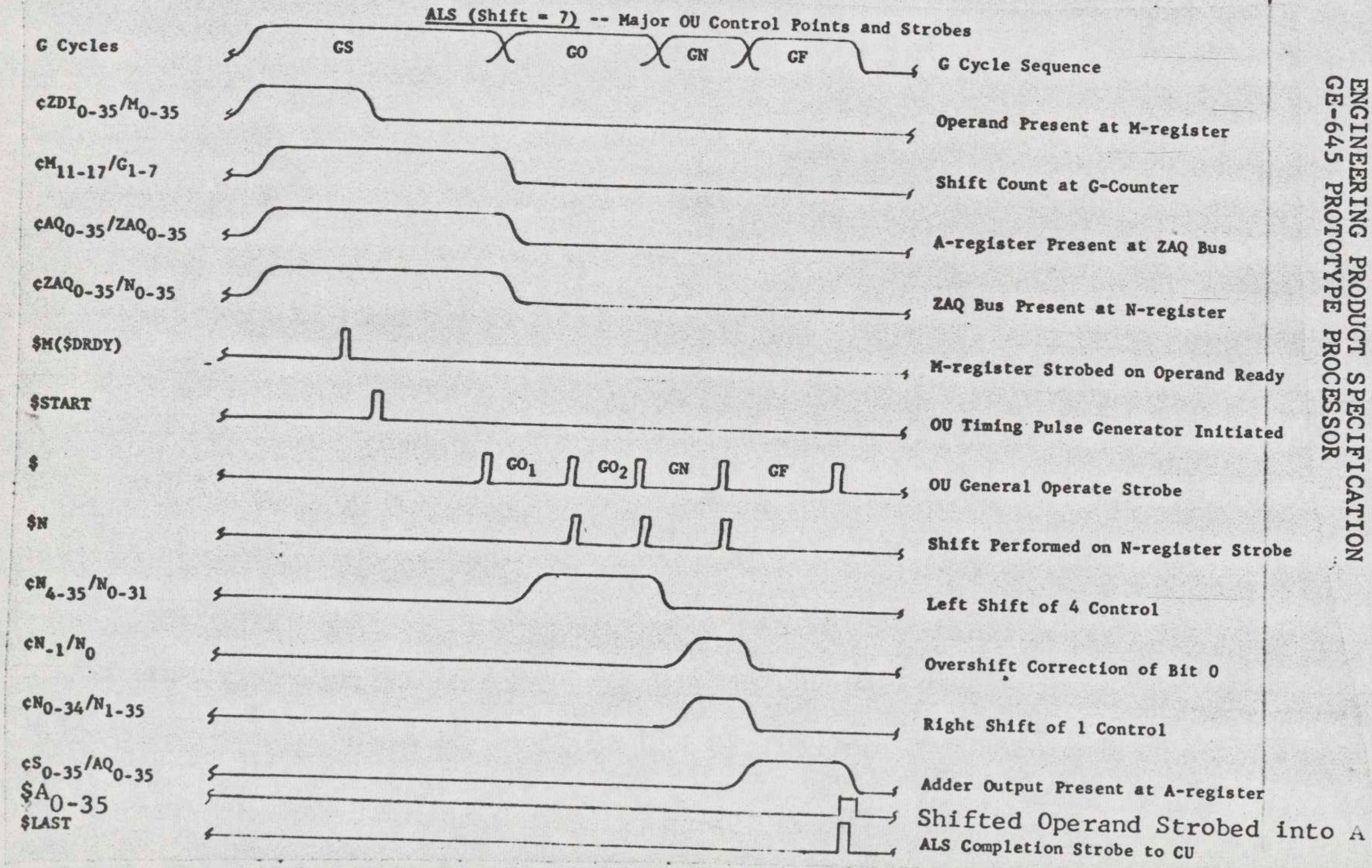

ALS (Shift = 7) -- Major OU Control Points and Strobes
G Cycles
GS
GO
GN
GF
G Cycle Sequence
¢ZDI0-35/M0-35
Operand Present at M-register
¢M11-17/G1-7
Shift Count at G-Counter
¢AQ0-35/ZAQ0-35
A-register Present at ZAQ Bus
¢ZAQ0-35/N0-35
ZAQ Bus Present at N-register
$M($DRDY)
M-register Strobed on Operand Ready
$START
OU Timing Pulse Generator Initiated
$
GO1
GO2
GN
GF
OU General Operate Strobe
$N
Shift Performed on N-register Strobe
¢N4-35/N0-31
Left Shift of 4 Control
¢N-1/N0
Overshift Correction of Bit 0
¢N0-34/N1-35
Right Shift of 1 Control
¢S0-35/AQ0-35
Adder Output Present at A-register
$A0-35
Shifted Operand Strobed into A
$LAST
ALS Completion Strobe to CU

Fig. 12. FIG. 12.3.4a ALS Instruction, Timing Diagram

12.3.4 - contd

Example 2: Left Rotate the AQ (LLR)

During this instruction the contents of the AQ-register are rotated out the left and in on the right by the amount specified in bits 11-17 of the LLR instruction word. The results are restored to AQ. For this example a rotation of 17 places will be executed.

ZDI/M controls and $ START generation are the same as for Example 1.

GS

When the $ START pulse propogates the medium delay elements the operate strobe $ occurs. This causes the following events:

1) The GS cycle to reset and the GO cycle is set.

2) $N strobes the contents of AQ into N.
3) $H strobes all zeros into H.
4) $G strobes the contents of M_{11-17} into G_{1-7} as shown:

	0	1	2	3	4	5	6	7
G =	0	0	0	1	0	0	0	1
	Shift 16				Shift 4		Shift 1	

5) The Zero and Sign Indicators are reset.

6) The $ strobe re-enters the Timing Pulse Generator to produce another $ strobe in approximately 200 nanoseconds.

12.3.4 - contd

(GO) (G = 17)

During this 200-nanosecond interval the first fast GO cycle is true and the control points are set up to execute the first rotation. The G contents call for one rotation of 16 and one rotation of 1. The greater rotations are executed first, so the rotation of 16 will occur during this first GO cycle. The control points enabled for this are ¢N_{0-15}/N_{56-71}, ¢N_{16-35}/N_{0-19}, ¢N_{36-51}/N_{20-35} and ¢N_{52-71}/N_{36-55}. This control will allow a left rotation on N by 16 positions on the next $N strobe.

On the $ strobe at the end of this first GO cycle, the following events occur:

1) $N strobes the N-register inputs and performs the rotation of 16.

2) $G decrements the G-counter by 16.

3) The GO cycle will be strobed but will not reset because the G count indicates another GO cycle is necessary.

4) The $ pulse will re-enter the Timing Pulse Generator to produce another $ in about 200 nanoseconds.

(GO) (G = 1)

During this second GO cycle the control is set up to execute the second rotation. At this time the G-counter has bit 7 true which calls for one rotation of 1. The control points enabled are ¢N_0/N_{71}, ¢N_{1-35}/N_{0-34}, ¢$(N_{36} + N_{36A}/N_{35})$ and ¢$N_{37-71}/N_{36/70}$. This control will allow a left rotation of N by 1 bit on the next $N strobe.

12.3.4 - contd

When the \$ strobe occurs at the end of the second GO cycle, the following events occur:

1) \$N strobes the N-register inputs and performs the rotation of 1.

2) \$G decrements the G-counter to zero by resetting G_7:

3) The GO cycle is reset and the GF cycle set.

4) The \$ strobe will re-enter the Timing Pulse Generator to procude another \$ in about 400 nanoseconds.

GF

At this point the rotation is complete and all that remains is to restore the rotated operand back to the AQ-register. This restore is done via the S-adder. The IAQ switch control enabled is $¢S_{0-35}/AQ_{0-35}$ and $¢S_{36-71}/AQ_{36-71}$. When the \$ strobe occurs at the end of GF, the following events occur:

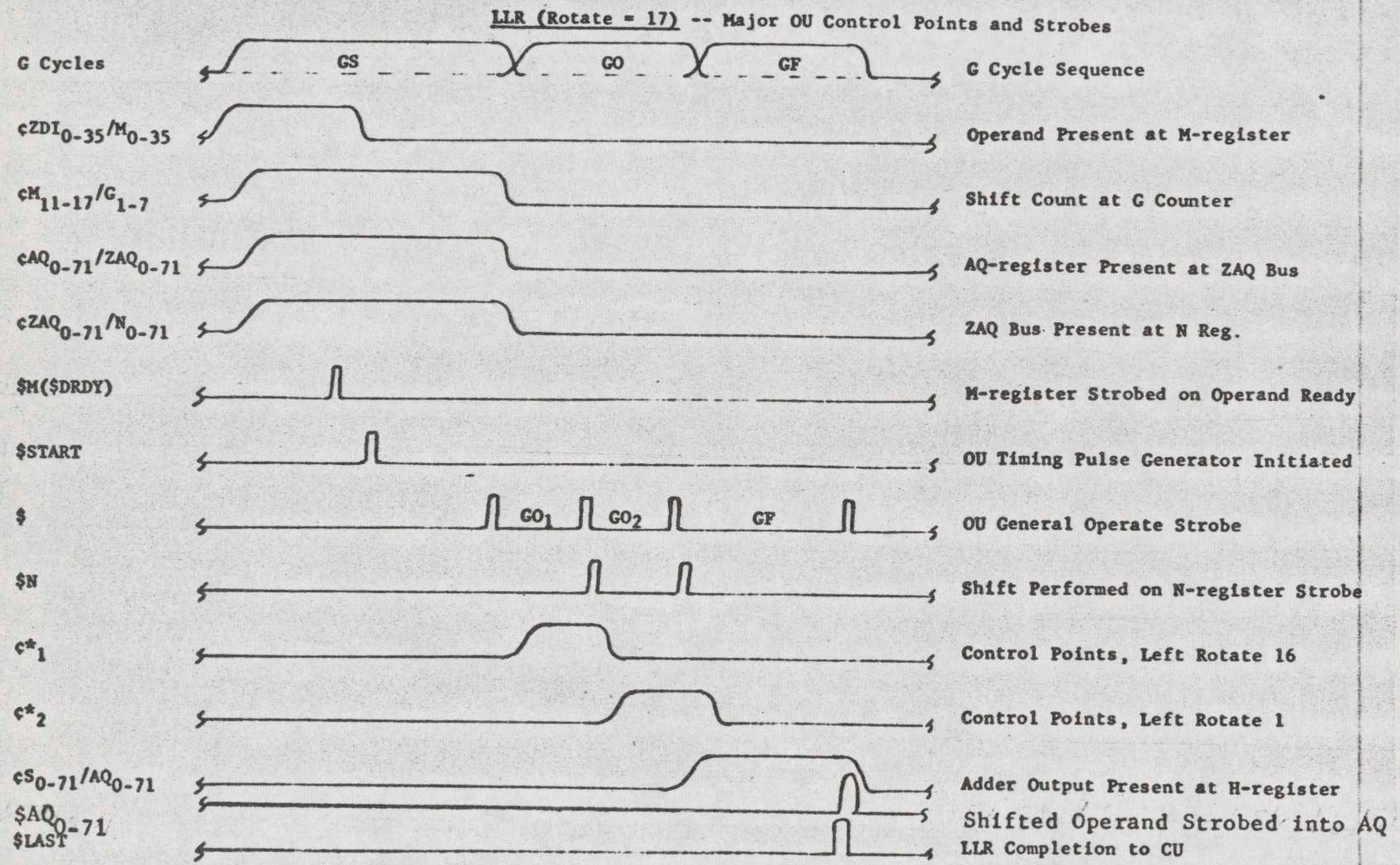
1) $\$AQ_{0-71}$ restores the rotated AQ operand.

2) The Zero Indicator is set if the contents of AQ = 0.

3) The Sign Indicator is set if bit zero of AQ = 1.

4) The GF cycle is reset.

5) It generates \$ LAST, which is sent to the CU to signal completion of the LLR instruction.


LLR (Rotate = 17) -- Major OU Control Points and Strobes

G Cycles
GS
GO
GF
G Cycle Sequence
¢ZDI0-35/M0-35
Operand Present at M-register
¢M11-17/G1-7
Shift Count at G Counter
¢AQ0-71/ZAQ0-71
AQ-register Present at ZAQ Bus
¢ZAQ0-71/N0-71
ZAQ Bus Present at N Reg.
$M($DRDY)
M-register Strobed on Operand Ready
$START
OU Timing Pulse Generator Initiated
$
GO1
GO2
GF
OU General Operate Strobe
$N
Shift Performed on N-register Strobe
¢*1
Control Points, Left Rotate 16
¢*2
Control Points, Left Rotate 1
¢S0-71/AQ0-71
Adder Output Present at H-register
$AQ0-71
Shifted Operand Strobed into AQ
$LAST
LLR Completion to CU

$*_1$ = $¢N_{0-15}/N_{56-71}$; $¢N_{16-35}/N_{0-19}$; $¢N_{36-51}/N_{20-35}$; $¢N_{52-71}/N_{36-55}$

$*_2$ = $¢N_{1-35}/N_{0-34}$; $¢N_{37-71}/N_{36-70}$; $¢N_0/N_{71}$; $¢(N_{36} + N_{36A}/N_{35})$

Fig 12.3

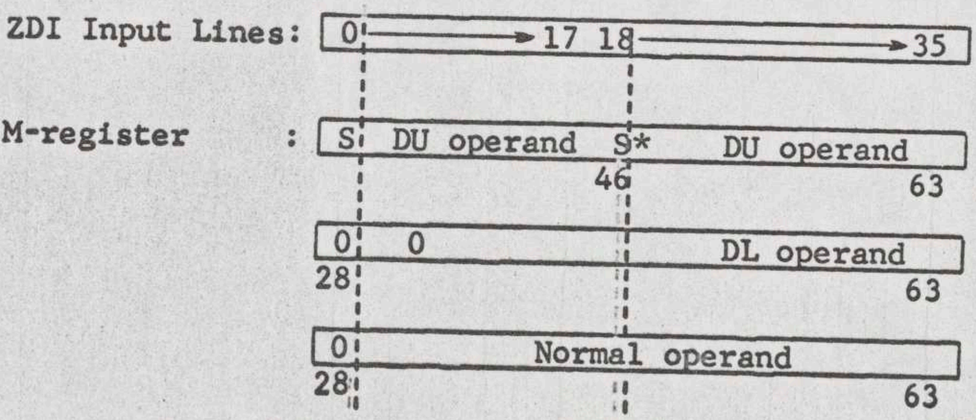
FIG. 12.3.4b LLR Instruction, Timing Diagram

12.3.5 FIXED POINT MULTIPLY GROUP

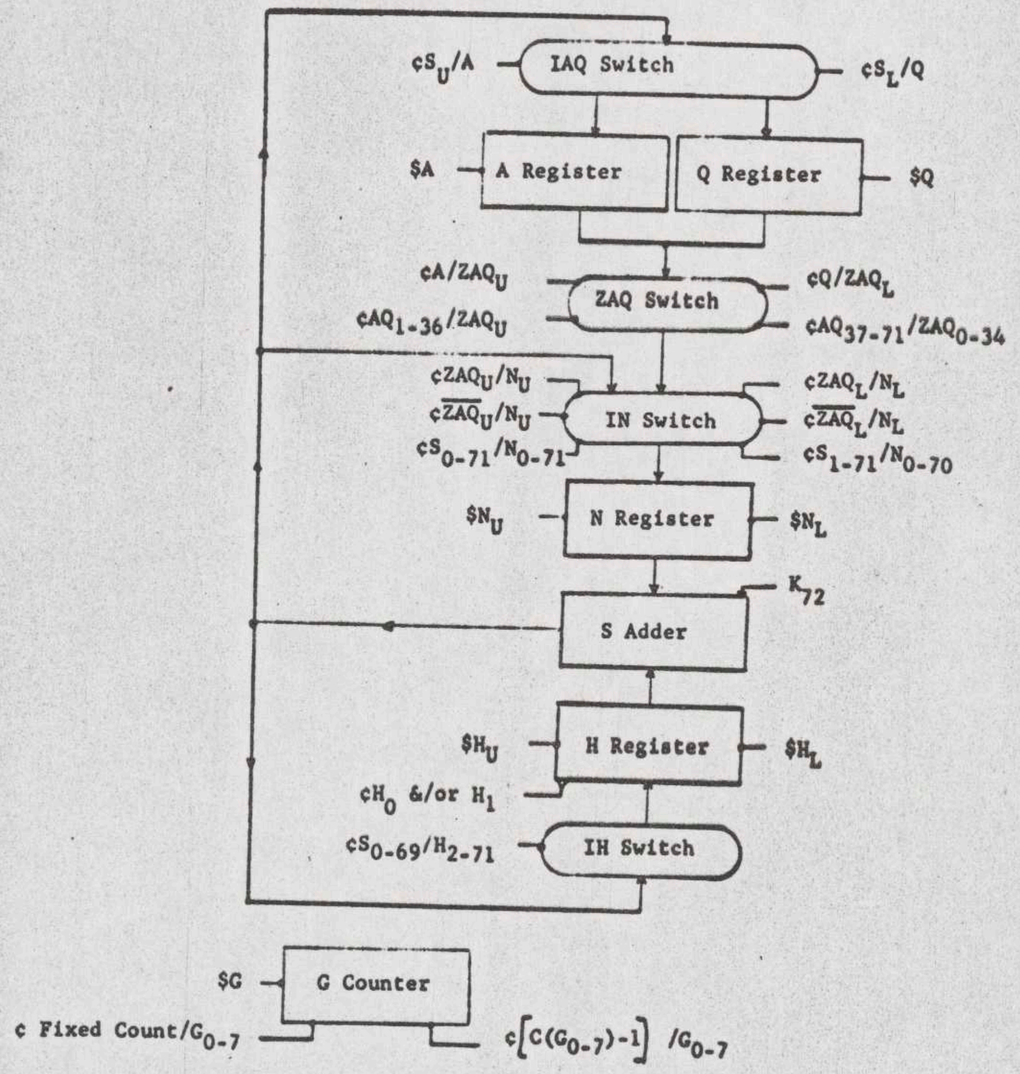
This group of two instructions provide the fixed point multiply capabilities of this machine. The major data paths and control points for these instructions are shown in Fig. 12.3.5.

The instructions MPY and MPF allow integer and fractional multiplication to be accomplished. For either instruction, a 36-bit (including sign) multiplicand is multiplied by a 36-bit (including sign) multiplier. When a Direct Upper (DU) operand is specified, the multiplier consists of the direct operand in bits 0-17 and zeros in bits 18-35. A Direct Lower (DL) modifier results in a multiplier with zeros in bits 0-17 and the DL operand in bits 18-35. The multiplicand is contained in the A-register for MPF and in the Q-register for MPY. For either instruction, the multiplier is contained in the M-register and is always right justified to bit 63 of the M-register. The partial products are accumulated in the H-register and the final 72-bit product is restored to the combined AQ-register.

The possible multiplier configurations which may be contained in the M-register with respect to the ZDI lines from the Base frame are shown below:



Major Data Paths and Control Points for the Fixed Point Multiply Instructions (OU)


IAQ Switch
¢S_U/A
¢S_L/Q
A Register
Q Register
$A
$Q
ZAQ Switch
¢A/ZAQ_U
¢AQ_{1-36}/ZAQ_U
¢Q/ZAQ_L
¢AQ_{37-71}/ZAQ_{0-34}
IN Switch
¢ZAQ_U/N_U
¢$\overline{ZAQ}_U$/N_U
¢S_{0-71}/N_{0-71}
¢ZAQ_L/N_L
¢$\overline{ZAQ}_L$/N_L
¢S_{1-71}/N_{0-70}
N Register
$$N_U$
$$N_L$
S Adder
K_{72}
H Register
$$H_U$
$$H_L$
¢H_0 &/or H_1
IH Switch
¢S_{0-69}/H_{2-71}
G Counter
$G
¢ Fixed Count/G_{0-7}
¢$[C(G_{0-7})-1]$ /G_{0-7}

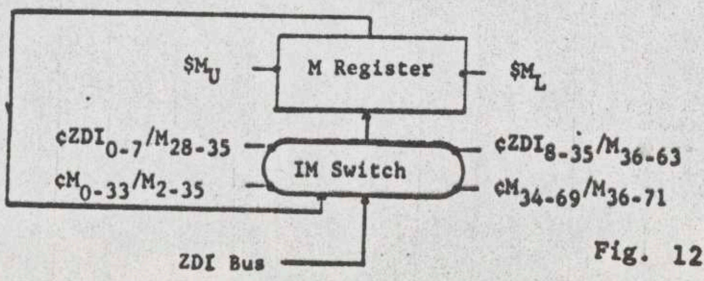

M Register
$$M_U$
$$M_L$
IM Switch
¢ZDI_{0-7}/M_{28-35}
¢M_{0-33}/M_{2-35}
¢ZDI_{8-35}/M_{36-63}
¢M_{34-69}/M_{36-71}
ZDI Bus

Fig. 12.3.5

12.3.5 - contd

When the multiplier is specified by a Direct Upper tag, the number of multiply cycles will be nine less than for those multipliers which are not Direct Upper. This results from the fact that for a Direct Upper multiplier the lower half of the 36-bit word is zero. The multiply algorithm used would result in nine passes or multiplications by zero for these 18 bits. To speed up the multiply in this case, the multiplications by zero are bypassed and only the meaningful 18 or Direct Upper bits are used to perform the multiply. For control convenience in the CU, a DU operand is placed on both the 0-17 and 18-35 output bit positions. This does not affect the OU because only bits 18-35 are used.

When the multiplier is specified by a Direct Lower tag, the upper 18 bits of a 36-bit word will be zero. No speed-up technique is employed in this case and the number of multiply cycles is 19. A Direct Lower multiplier will always be a positive number.

For the detailed control of both instructions in this group, refer to the OU Specification Chart 43D140232, page 8.

In the explanation of all the example instructions, the following conditions will be assumed:

1) The instruction has been strobed into the 0 register.

2) The GS cycle flip-flop has been set.

3) The DDF flip-flop has been set indicating the operation decode has settled.

NOTE: Reference to Fig. 12.3.5 and the partial timing diagram at the end of the example will aid in the understanding of the instruction execution and implementation.

12.3.5 - contd

Example 1: Fractional Multiply (MPF)

During this instruction, the contents of the A-register are multiplied by the contents of a memory location Y to produce a 71-bit fractional product plus sign which is stored in the AQ-register. Bit position 71 of the AQ product will contain a zero. For this example, no operands are used but all major controls are explained.

The IM switch is enabled by the presence of the MPF decode together with DIC and DPFF reset. The control points ¢ZDI_{0-7}/M_{28-35} and ¢ZDI_{8-35}/M_{36-63} are enabled to allow the ZDI bus to be present at the M-register inputs. When the operand ready strobe $DRDY is received from the CU, the Double Precision flip-flop not being set allows the M-register strobe $\$M_{0-71}$ to be generated and strobe the multiplier into the M-register. See page 12-104 for the possible arrangements of the multiplier. If the multiplier operand being furnished by the CU is a Direct Upper, $DMF will be received from the CU at the same time as $DRDY. $DMF will set the Direct Upper Multiplier flip-flop (DUM). The $DRDY strobe is turned around as $YRY and returned to the CU to signal that a new instruction may be prepared. $YRY also sets the operand ready flip-flop, and with the MPF decode sets the M-register is busy with a multiply flip-flop (DIC). The DIC flip-flop is used as an inhibit on the ZDI/M controls for the duration in which the multiplier is contained in the M-register.

GS

During the 320 nanoseconds of GS form $ START to the first $, the control is established to load the G-counter with the proper count on the first $. If DUM is set (which defines the multiplier as a DU, Fixed Count/G control will force a count of 10 into G (that is, set G_4 and G_6). If DUM is reset, a count of 19 is placed in G (that is, set G_3, G_6, and G_7).

12.3.5 - contd

As per the multiply algorithm, the low-order two bits of the multiplier M_{62-62} are decoded during this same 320 nanoseconds of GS to indicate whether the first mul multiplication will be by 0, 1, 2, or 3. The decode of M_{62-63} is interpreted as:

1) If M_{62-63}=00, this represents multiplication by zero or a pass condition (that is, no modification of the partial product in H which at this time is zero). No ZAQ or IN switch is enabled.

2) If M_{62-63}=01, this represents multiplication of the multiplicand by one. Since at this time there is no partial product, the multiplicand will become the first partial product. To accomplish this, the ZAQ and IN switches are enabled by ¢AQ_{0-35}/ZAQ_{0-35} and ¢ZAQ_{0-35}/N_{0-35}. These enabled switches allow the A-register multiplicand to be present at the N-register inputs.

3) If M_{62-63}=10, this represents multiplication of the multiplicand by two. For a binary number this may be done by shifting the number left by one bit position. To accomplish this, the ZAQ and IN switches are enabled by ¢AQ_{1-36}/ZAQ_{0-35} and ¢ZAQ_{0-35}/N_{0-35}. The control input to N_{35} is forced to be a zero at this time so that the effective AQ/N transfer will be AQ_{1-35}/N_{0-34}.

4) If M_{62-63}=11, this represents multiplication of the multiplicand by three. This is accomplished by doing a multiply of (4-1=3). The multiply by 4 is accomplished by adding one to the bit which will be contained in M_{63} during the next multiply cycle. Effectively, this means that the bits presently contained in M_{60-61} are to be incremented by one. The minus one control is set up during the present multiply cycle (GS) and is performed by subtracting the contents of A from the partial product. The ZAQ and IN switches are enabled by ¢AQ_{0-35}/ZAQ_{0-35}, dZAQ_{0-35}/N_{0-35} ¢$\overline{ZAQ}_{36-71}/N_{36-71}$. These switches present the one's complement of the A-register to N_{0-35} and all ones are present at the N_{36-71} inputs.

12.3.5 - contd

Because a new set of multiplier bits will be needed for the next multiply cycle (the first GO cycle), the control is set up to allow a right shift of the M-register by two. The IM switch is enabled by ¢M_{0-33}/M_{2-35} and ¢M_{34-69}/M_{36-71} to accomplish this right shift.

When the $ START pulse propagates the medium delay elements, the first operate strobe $ occurs. This allows the following events:

1) The GS cycle to reset and the GO cycle to set.

2) The Zero and Sign Indicators to reset.

3) $G is generated which strobes the count of 10 or 19 into the G-counter depending on the control lines enabled.

4) $M is generated which strobes the M-register and executes a right shift of two to set up a new set of multiplier bits.

5) $H is generated which strobes the H-register, but since no IH switch is enabled at this time, all zeros are loaded into H.

6) $N is generated which strobes the N-register and sets up the data to produce the first partial product in the adder. The data loaded into N depends on which controls were enabled from A into N as a function of the first set of M_{62-63} bits.

7) The K_{72} adder control flip-flop and String flip-flop (STFF) are set if the first M_{62-63} set were 11_2 which called for a multiplier of 3.

8) The PASS flip-flop is set if the first M_{62-63} set were 00_2.

12.3.5 - contd

9) The Partial Product Sign flip-flop (PPS) has the contents of A_0 strobed into it if a multiplier of 1 or 2 was called for, has $\overline{A}_0$ strobed in if the multiplier were 3, or is left unchanged if the multiplier was zero.

10) The $ strobe re-enters the Timing Pulse Generator delay line network to produce another $ strobe in about 320 nanoseconds.

(GO) G > 3)

During this 320 nanosecond time the first GO cycle is true and the S-adder is used to produce the first partial product and make it available on the S bus at the adder output. While this partial product is being developed, the control points are established to store the partial product on the S bus into H and set up the control to perform the next 2-bit multiplication.

During the GO cycles, the determination of what the next effective multiplier will be is not only a function of M_{62-63} as it was in GS, but it is also dependent on the state of the String flip-flop (STFF). Although there are still only four possibilities for a multiplier, there are now two conditions that cause each possibility. These are shown in truth table form:

M_{62}	M_{63}	STFF	Effective Multiplier
0	0	0	0
0	1	0	+1
1	0	0	±2
1	1	0	3 (or -1)
0	-	1	+1
0	1	1	+2
1	0	1	3 (or -1)
1	1	1	0

12.3.5 - contd

The control points enabled as a function of the effective multiplier are the same as the ones mentioned during GS when the multiplier was 0, 1, 2, or 3.

The IH switch is enabled during GO to allow the partial product to be strobed into H. The control points for this during GO are ¢S_{0-33}/H_{2-35} and ¢S_{34-69}/H_{36-71}.

During this GO cycle, the controls on the IM switch are again enabled to allow the M-register to be right shifted by two in order to have a new set of multiplier bits for the next cycle. The Full Counter Decrement control (¢FCD) becomes enabled so that the next $G will decrement the count by one.

When the first $ strobe propagates the medium delay elements, the second $ is distributed to the OU to end the first GO cycle and the following events occur:

1) The GO flip-flop is strobed, but because the G count is not zero it remains set.

2) $G decrements the G-counter by one.

3) $M strobes the M-register and sets in a new pair of multiplier bits.

4) $H strobes the partial product into H_{2-71}. The content of the Partial Product Sign flip-flop (PPS) is strobed into H_0 unconditionally and into H_1 if no overflow occurred.

12.3.5 - contd

5) \$N strobes the new premultiplied multiplicand in the N-register as determined by the multiplier decode.

6) The K_{72} adder control flip-flop and the String flip-flop (STFF) are set if the effective multiplier is 3. STFF is reset if the effective multiplier is +1, +2, or ±2 and not changed if it is zero. K_{72} is reset for values of other than 3.

7) The PASS flip-flop is set for an effective multiplier of zero, or reset for 1, 2, or 3.

8) The PPS flip-flop is strobed and sets or resets for the same conditions listed at the end of GS.

9) The \$ strobe re-enters the Timing Pulse Generator delay line network to produce another \$ strobe in about 320 nanoseconds.

Up until the time when the G-counter is decremented to a count of 3, the functions performed during all of the GO cycles are iterative of the first GO cycle.

12.3.5 - contd

(GO) (G = 3)

During the GO cycle when G = 3, the next to last partial product is formed in the S-adder, and the last multiplicand is prepared for addition to the partial product during G = 2 time. The conditions to enable control points during G = 3 are the same as those used for G > 3 with these exceptions:

1) For the decode of $M_{62-63} = 10_2$ and STFF = 0 (multiplier = ±2), two times the multiplicand is prepared for subtraction from the partial product during G = 2. This necessitates that $¢\overline{ZAQ}_{0-35}/N_{0-35}$ and $¢\overline{ZAQ}_{36-71}/N_{36-71}$ be enabled and the control to set K_{72} must be enabled.

2) When G = 3, a level resets the DIC flip-flop and frees the M-register.

3) The control on the set/reset of PPS is not a function of the (±2) multiplier.

4) The set/reset control of STFF is not a function of any multiplier.

When the $ pulse is generated at the end of G = 3, the same strobes occur as for G > 3 with the exception of $M. $M is not produced because the multiplier in M has been exhausted by this time. The String flip-flop (STFF) is unconditionally reset at this time. This strobe decrements the G count to 2 and the multiply continues.

12.3.5 - contd

(GO) (G = 2)

During G = 2, the final partial product is formed in the S-adder as a function of the former partial product in H and the multiplier controlled multiplicand in N. Because this will be the last add cycle, no control points are enabled to load a new multiplicand into N on the strobe at the end of G = 2.

When the \$ pulse is generated at the end of G = 2, the same strobes are produced that occurred at the end of G = 3. Because no switches are enabled into N, this register is cleared to all zeros. The K_{72} flip-flop is unconditionally reset and the G count is decremented to one.

nditionally reset and the G

(GO) (G = 1)

During G = 1 the final fractional product is contained in the H_{1-71} register with the binary point to the right of H_1. The control is set up to transfer H_{1-71} to N_{0-70} and thereby place the binary point to the right of N_0. The control point enabled for this is ¢S_{1-71}/N_{0-70}. When the \$ pulse is generated at the end of G-1, the following events occur:

1) The GO cycle is reset and the GF cycle set.

2) \$N transfers the product with a left shift of one from H via S adder to N. N_{71} is cleared to zero.

3) \$G decrements the G count to zero.

4) The Sign Indicator is set if $S_0 = 1$.

5) The Zero Indicator is set if the adder output is zero.

6) The Overflow Indicator is set if bit S_0 and S_1 are different.

7) The \$ strobe re-enters the Timing Pulse Generator delay line network to produce the last \$ strobe in about 400 nanoseconds.

12.3.5 - contd

GF

During GF the control points are enabled to transfer the contents of N to AQ. This is done via the adder and IAQ switch with $¢S_{0-35}/AQ_{0-35}$ and $¢S_{36-71}/AQ_{36-71}$.

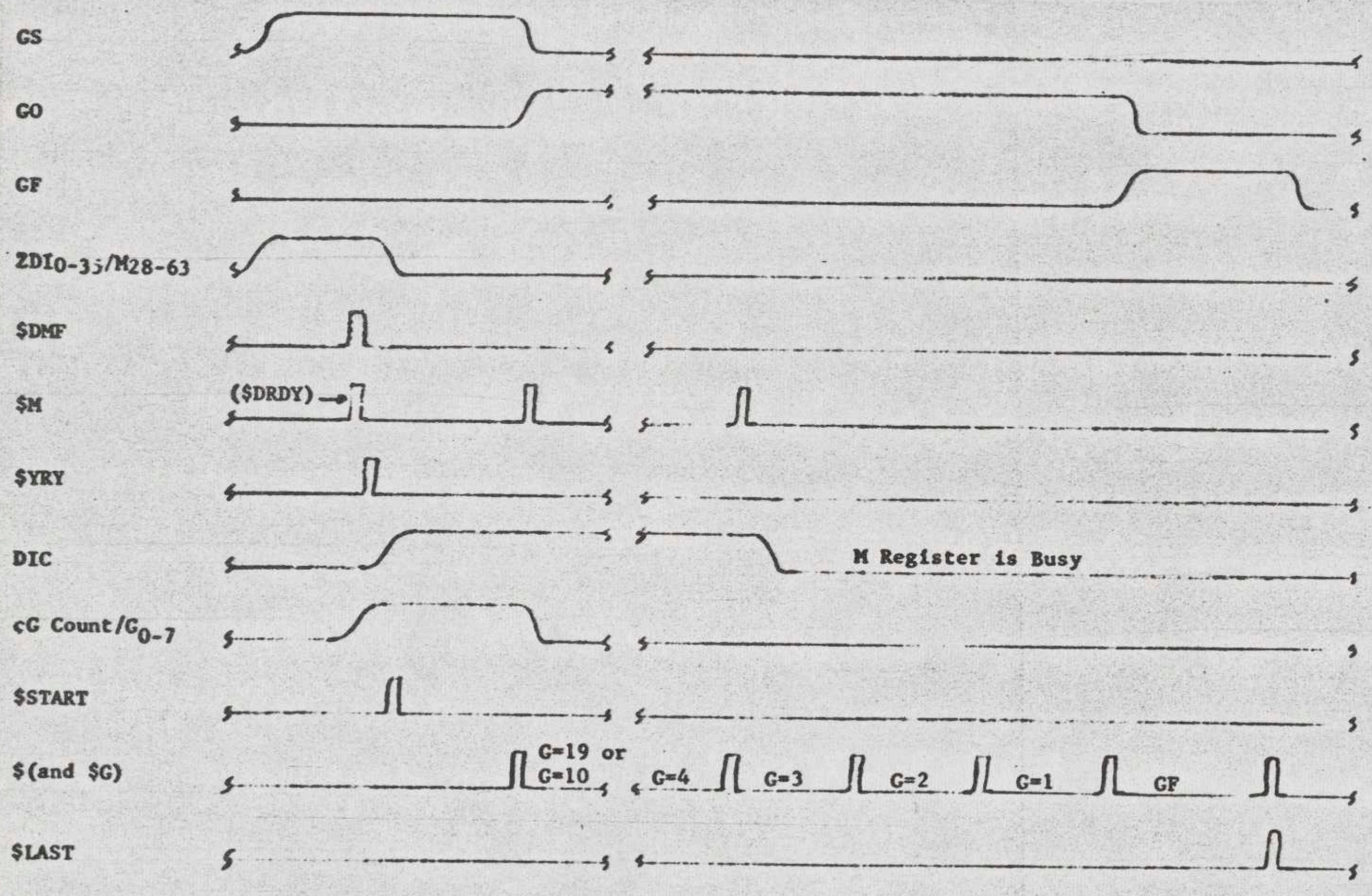
When the \$ strobe occurs, the following events occur:

1) The GF cycle is reset.

2) The product is strobed into AQ.

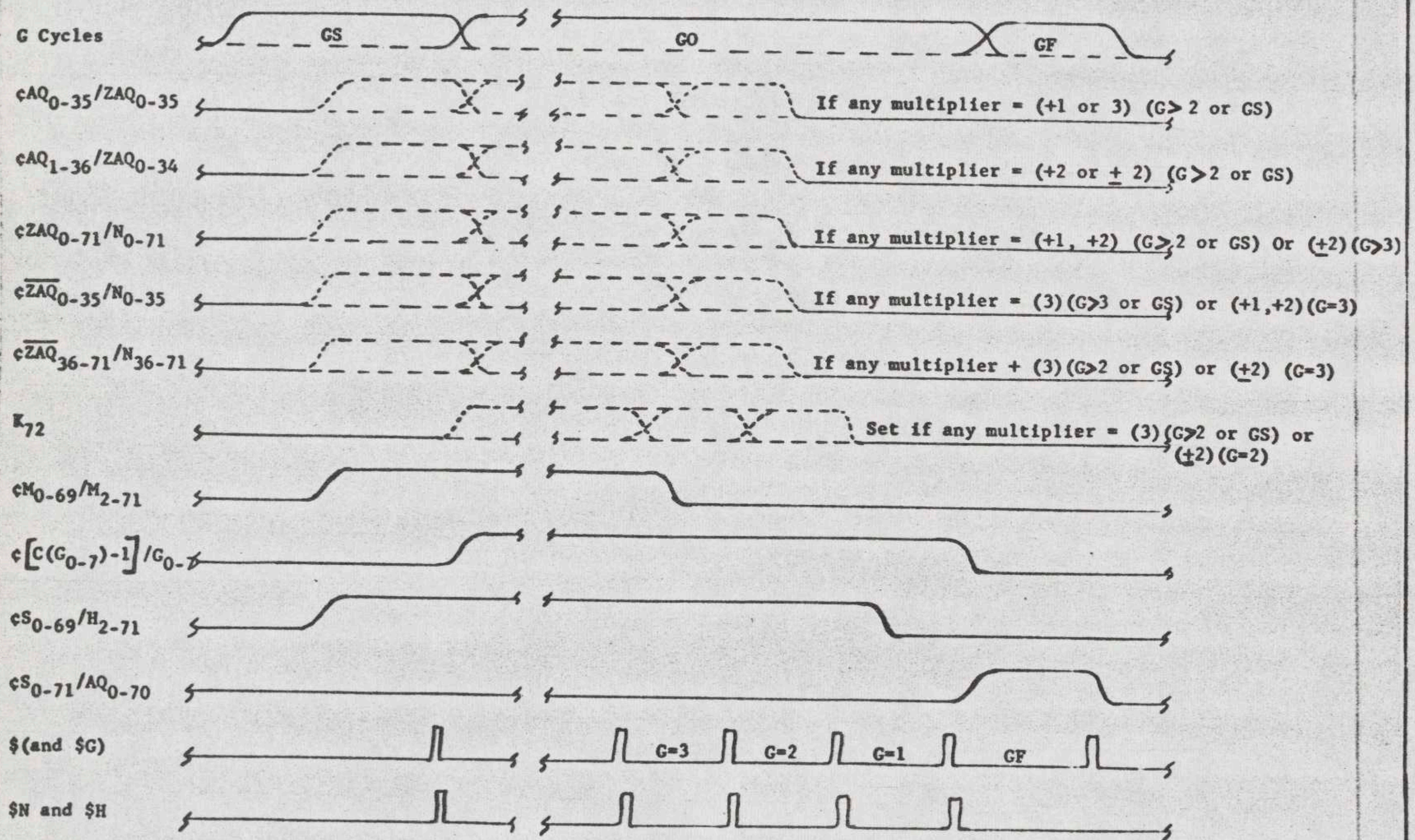
3) It generates \$ LAST, which is sent to the CU to signal completion of the MPF instruction.

The execution of MPY is the same as MPF except that the multiplicand is contained in the Q-register, and during the G = 1 or product alignment cycle, the product is already correctly aligned with the binary point to the right of H_{71}. Therefore, the contents of H_{0-71} is transferred directly to N_{0-71} via the adder with no shift.


GS
GO
GF
ZDI0-35/M28-63
$DMF
$M
($DRDY)
$YRY
DIC
M Register is Busy
cG Count/G0-7
$START
$(and $G)
G=19 or
G=10
G=4
G=3
G=2
G=1
GF
$LAST

MPF - - Major OU Control Points and Strobes (Part 1)

FIG. 12.3.5a MPF Instruction, Timing Diagram
(Sheet 1 of 2)


G Cycles
GS
GO
GF
$¢AQ_{0-35}/ZAQ_{0-35}$
If any multiplier = (+1 or 3) (G> 2 or GS)
$¢AQ_{1-36}/ZAQ_{0-34}$
If any multiplier = (+2 or ± 2) (G>2 or GS)
$¢ZAQ_{0-71}/N_{0-71}$
If any multiplier = (+1, +2) (G>2 or GS) Or (±2)(G>3)
$¢\overline{ZAQ}_{0-35}/N_{0-35}$
If any multiplier = (3)(G>3 or GS) or (+1,+2)(G=3)
$¢\overline{ZAQ}_{36-71}/N_{36-71}$
If any multiplier + (3)(G>2 or GS) or (±2) (G=3)
K_{72}
Set if any multiplier = (3)(G>2 or GS) or (±2)(G=2)
$¢M_{0-69}/M_{2-71}$
$¢[C(G_{0-7})-1]/G_{0-7}$
$¢S_{0-69}/H_{2-71}$
$¢S_{0-71}/AQ_{0-70}$
$(and $G)
G=3
G=2
G=1
GF
$N and $H

MPF - - Major OU Control Points and Strobes (Part 2)

Fig 12-3.5A

FIG. 12.3.5a MPF Instruction, Timing Diagram
(Sheet 2 of 2)

12.3.6 FIXED POINT DIVIDE GROUP

This group of two instructions provides the fixed point divide capabilities of this machine. The major data paths and control points for these instructions are shown in Fig. 12.3.6. The instructions DIV and DVF allow integer and fractional division to be accomplished. A 36-bit divisor is divided into a 36-bit dividend for DIV or into a 71-bit dividend for DVF to produce a 36-bit quotient plus a 36-bit remainder. The dividend is contained in Q for DIV or AQ for DVF, the divisor is contained in H after the initial load, and the quotient and remainder are stored in the AQ-register.

DVF

Whenever performing fractional division the dividend and divisor must be less than unity and the resulting quotient must also be less than unity. Division by zero also is an invalid operation.

$\frac{\cdot X}{\cdot Y} \geq 1$ not valid

$\frac{\cdot X}{\cdot Y} \leq 1$ is valid

$\frac{\cdot X}{0 - 0}$ not valid

Detection of either of two invalid magnitude conditions will immediately abort the divide execution and result in an entry into the Divide Check Fault routine.

A quotient of $+2^{35}$ is not permitted since Q-register capacity is $(2^{35}-1)$. Although a quotient of -2^{35} could be represented with the 36-bit word size, for hardware simplification this is not allowed.

The data format for some of these quantities are:

Major Data Paths and Control Points for the Fixed Point Divide Instructions (OU)

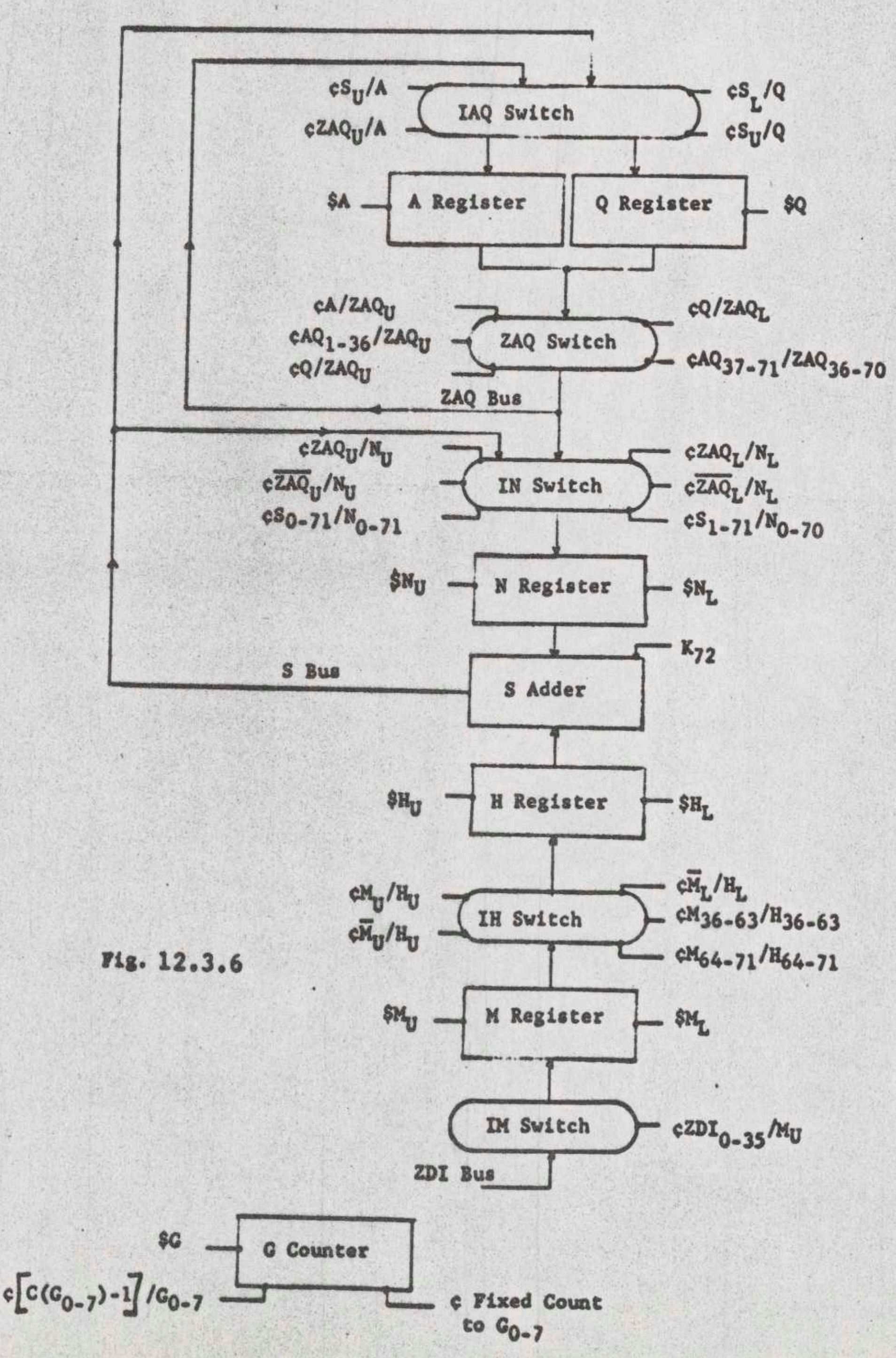
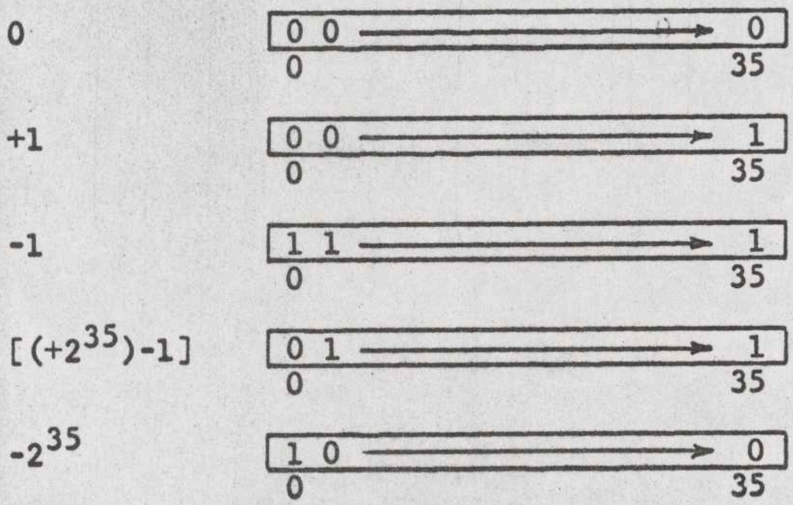

¢S_U/A
¢ZAQ_U/A
IAQ Switch
¢S_L/Q
¢S_U/Q
\$A
A Register
Q Register
\$Q
¢A/ZAQ_U
¢AQ_{1-36}/ZAQ_U
¢Q/ZAQ_U
ZAQ Switch
¢Q/ZAQ_L
¢AQ_{37-71}/ZAQ_{36-70}
ZAQ Bus
¢ZAQ_U/N_U
¢$\overline{ZAQ}_U$/N_U
¢S_{0-71}/N_{0-71}
IN Switch
¢ZAQ_L/N_L
¢$\overline{ZAQ}_L$/N_L
¢S_{1-71}/N_{0-70}
\$$N_U$
N Register
\$$N_L$
K_{72}
S Bus
S Adder
\$$H_U$
H Register
\$$H_L$
¢M_U/H_U
¢$\overline{M}_U$/H_U
IH Switch
¢$\overline{M}_L$/H_L
¢M_{36-63}/H_{36-63}
¢M_{64-71}/H_{64-71}
\$$M_U$
M Register
\$$M_L$
IM Switch
¢ZDI_{0-35}/M_U
ZDI Bus
\$G
G Counter
¢[C(G_{0-7})-1]/G_{0-7}
¢ Fixed Count to G_{0-7}

Fig. 12.3.6

12.3.6 - contd



The divide algorithm used for these instructions makes it necessary that the dividend always be positive and the divisor always be negative before the division is executed. The operands are converted, when necessary, prior to entering the operate (GO) cycles.

To illustrate the control used in the execution of these instructions, one (DIV) will be explained in detail. For the detailed control of both instructions in this group, refer to the OU Specification Chart 43D140232, page 9.

12.3.6 - contd

In the explanation of the example instruction, the following conditions will be assumed:

1) The instruction code has been strobed into the O-register.

2) The GS cycle flip-flop has been set.

3) The DDF flip-flop has been set indicating the operation decode has settled.

NOTE: Reference to Fig 12.3.6 and the partial timing diagram at the end of the example will aid in understanding of the instruction execution and implementation.

Example 1: Divide Integer (DIV)

During this instruction a 35-bit integer dividend plus sign in the Q-reigster is divided by a 35-bit integer divisor plus sign from a memory location Y to produce a 35-bit quotient plus sign and a 35-bit remainder plus sign. The contents of the A-register are destroyed by this instruction as a result of the remainder being stored in A at the completion of the instruction.

The load path for the divisor operand into the M-register places ZDI_{0-35} into M_{0-35}. The generation of $ SET GS and $ START are as developed in paragraph 12.2.

GS

During GS, the ZAQ and IN switches are enabled to convert C(Q) to a positive dividend. If the sign of the dividend in Q is already positive ($Q_0 = 0$), the control points activated are ¢AQ_{36-71}/ZAQ_{0-35}, ¢ZAQ_{0-35}/N_{0-35}, and

ZAQ

12.3.6 - contd

¢ZAQ_{36-71}/N_{36-71}. If the sign of the dividend is negative ($Q_0 = 1$), the control points ¢AQ_{36-71}/ZAQ_{0-35}, ¢$\overline{ZAQ}_{0-35}/N_{0-35}$, ¢$\overline{ZAQ}_{36-71}/N_{36-71}$, and ¢ SET K_{72} are enabled to complement Q and convert the dividend to positive.

When the $ START pulse propagates the medium delay elements, the first operate strobe $ occurs. This allows the following events:

1) The GS cycle to reset and the GD1 cycle to set.
2) The Quotient Sign flip-flop (QSF) and Zero Indicator to reset.
3) $N strobes either ZAQ or $\overline{ZAQ}$ into the N-register.
4) $H is generated, but since no IH switch is enabled, all zeros are loaded into H.
5) The K_{72} adder control flip-flop is set if it is necessary to complement Q to produce a positive dividend.
6) The Sign Indicator is set if Q_0 - 1 or reset if $Q_0 = 0$.
7) The $ strobe re-enters the Timing Pulse Generator delay line network to produce another $ strobe in about 400 nanoseconds.

12.3.6 - contd

GD1

During GD1 the positive dividend will become available at the adder output. Switches are enabled during this time to restore the positive dividend into Q, clear A to zero and set up conditions for a negative divisor to be available during GD2 and the GO cycles.

The IAQ switch is enabled by $¢S_{0-35}/AQ_{36-71}$. The IH switch is enabled by $¢M_{0-35}/H_{0-35}$ if $M_0 = 1$ or by $¢M_{0-35}/H_{0-35}$ and $¢\overline{M}_{36-71}/H_{36-71}$ if $M_0 = 0$. When the $ strobe occurs at the end of GD1, the following events take place:

1) The GD1 cycle is reset and GD2 cycle is set.

2) $N is generated, but since no IN switch is enabled, all zeros are loaded into N.

3) $H is generated transferring the divisor into H.

   If the original divisor was positive, the one's complement of M is strobed into H at this time. The negative divisor is obtained by leaving K_{72} set and the one's complement of M in H until G = 2 time. For this case, the H-register does not contain the full negative divisor.

4) $AQ is generated which loads a positive dividend into Q and clears A to all zeros.

5) The K_{72} adder control flip-flop is set if it is necessary to complement M to produce a negative divisor. K_{72} will remain set until G = 2.

6) The Quotient Sign flip-flop (QSF) is set if the signs of the original dividend and divisor were different.

7) The $ strobe re-enters the Timing Pulse Generator delay line network to produce another $ in about 400 nanoseconds.

12.3.6 - contd

GD2

During GD2 the negative divisor will be produced in the adder and become available on the S bus. Switches are enabled during this time to transfer the positive dividend with a left shift of one from AQ to N, set the G-counter to a count of 38, set the Compare Magnitude flip-flop for the compare of the dividend and divisor during GO, and set the Zero Indicator and Divide Check Fault trap if the divisor is zero.

The ZAQ and IN switches are enabled by ¢AQ$_{1-36}$/ZAQ$_{0-35}$, ¢AQ$_{37-71}$/ZAQ$_{36-70}$, ¢ZAQ$_{0-35}$/N$_{0-35}$, and ¢ZAQ$_{36-71}$/N$_{36-71}$. This left shift of one of the dividend is done so that only one magnitude comparison is done in order to determine whether the quotient will be greater than $(2^{35}-1)$ and therefore not representable with a 35-bit word plus sign.

When the $ strobe occurs at the end of GD2, the following events take place:

1) The GD2 cycle to reset and the GO cycle to set.

2) $N is generated which strobes the divisor left shifted by one into N.

3) $G is generated which strobes the count of 38 into the G-counter.

4) The Zero Indicator is set and the Divide Check Fault strobe ($DVCK) is sent to the CU if the negative divisor is zero. The Abort flip-flop is set in the OU and the OU is reset to an initialized state.

5) The Compare Magnitude flip-flop (CMFF) is set.

6) If an Abort did not occur because of a zero divisor, the $ strobe re-enters the Timing Pulse Generator to produce another $ strobe in about 400 nanoseconds.

12.3.6 - contd

(GO) (CMFF)

During this 400 nanosecond period the first GO cycle is true. The magnitude of the divisor and dividend are compared in the S-adder at this time to determine if a valid quotient will be produced as a result of the DIV execution. If a carry is generated out of bit zero of the adder, then a Divide Check Fault will occur and the divide aborted. Also, during this cycle the control is established to prepare a dividend for the first divide operation which will occur in the second GO cycle if the divide trap does not get set (that is, $SC_0 = 0$). The control points enabled for this are $¢N_{1-35}/N_{0-34}$, $¢N_{36}/N_{35}$, and $¢N_{37-71}/N_{36-71}$. This control will allow a left shift of the dividend by one bit which will produce a dividend to guarantee that the final quotient will not be greater in magnitude than $(2^{35}-1)$.

The fixed count decrement (¢FCD) lines are enabled at this time to allow the G-counter to be decremented by one on the next strobe.

When the $ strobe occurs at the end of the first GO cycle, the following events occur:

1) The GO cycle is strobed, but since the G count is not zero it is not reset.

2) The Compare Magnitude flip-flop is reset.

3) The Divide Check Fault strobe ($DVCK) is sent to the CU if a carry is generated out of bit zero of the adder (that is, $SC_0 = 1$). The Abort flip-flop is set in the OU and the OU is reset to an initialized state.

4) $N is generated which initiates the left shift of the dividend by one.

5) $G decrements the G count to 37.

6) $AQ is generated if there is no carry from bit zero of the adder and clears the AQ-register to zeros.

7) If an Abort did not occur, the $ strobe re-enters the Timing Pulse Generator to produce another $ in about 320 nanoseconds.

12.3.6 - contd

(GO)(G > 3)

During the next 34 GO cycles or until the G count equals 3, all but the last of the divide subcycles is performed to produce the quotient. For each of these 34 GO cycles, the contents of the dividend in N and the divisor in H are subtracted in the S-adder. At the end of each cycle a test is made on the carry from bit zero of the adder. If there is a carry (that is, remainder ≥ divisor), then the adder result is shifted left one bit and transferred to the N-register to form a new dividend for the next divide.operation. The control points enabled for this are ¢S_{1-36}/N_{0-35} and ¢S_{37-71}/N_{36-70}. If there is no carry from bit zero of the adder (that is, remainder < divisor), then the contents of the N-register are shifted left one bit to form a new dividend for the next divide operation. The control points enabled for this are ¢N_{1-35}/N_{0-34}, ¢N_{36}/N_{35}, and ¢N_{37-71}/N_{36-70}.

At the end of each of the 34 GO cycles, the contents of the AQ-register are shifted left one bit and carry from bit zero of the adder is transferred to bit 35 of the A-register. This allows the quotient to be generated on a one bit basis with the most significant bit being produced first. The control points enabled for this are ¢AQ_{1-36}/ZAQ_{0-35}, ¢AQ_{37-71}/ZAQ_{36-70}, ¢ZAQ_{0-35}/AQ_{0-35}, and ¢SC_0/AQ_{35}.

When the $ strobe occurs at the end of each of the 34 GO cycles, the following events occur:

1) $N loads a new dividend into the N-register depending on the state of the carry from the adder bit zero.

2) $A shifts the partial quotient left one bit and loads in a new quotient bit.

3) $G decrements the G-counter by one.

4) The $ strobe re-enters the Timing Pulse Generator to produce a subsequent $.

12.3.6 - contd

(GO (G = 3)

The above procedure is repeated until the G-counter equals 3. When G = 3, the same events occur as for G > 3 except that if the carry from bit zero of the adder is one, the adder result is transferred to N with no left shift, and if the carry was zero, the contents of N are not changed. Also, the H-register is cleared to zero at the end of (GO) (G = 3). The control points enabled if the adder carry is one are $¢S_{0-35}/N_{0-35}$ and $¢S_{36-71}/N_{36-71}$.

When the $ strobe occurs at the end of (GO)(G = 3), the same events occur as for G > 3 except:

1) $N is generated only if the carry from bit zero of the adder was a one.

2) $H clears the H-register.

3) The K_{72} adder control flip-flop is unconditionally reset.

4) The $ strobe re-enters the Timing Pulse Generator to produce a subsequent $.

12.3.6 - contd

(GO) (G = 2)

During (GO) (G = 2) the quotient and remainder are exchanged between the A-register and the N-register. The quotient is transferred into N in such a manner that the quotient sign may be connected if necessary. The transfer of the remainder in N to A is done via the adder by enabling $¢S_{0-35}/AQ_{0-35}$. The transfer of the quotient in the A-register to N is done as a function of the Quotient Sign flip-flop (QSF) which may have been set in GD1. If QSF is set, the control is established to transfer the one's complement of AQ to N and set K_{72}. The control points enabled are $¢AQ_{0-35}/ZAQ_{0-35}$, $¢AQ_{36-71}/ZAQ_{36-71}$, $¢\overline{ZAQ}_{0-35}/N_{0-35}$ and $¢\overline{ZAQ}_{36-71}/N_{36-71}$. If QSF is not set, AQ is transferred straight through into N via $¢AQ_{0-35}/ZAQ_{0-35}$, $¢AQ_{36-71}/ZAQ_{36-71}$, $¢ZAQ_{0-35}/N_{0-35}$, and $¢ZAQ_{36-71}/N_{36-71}$.

When the \$ strobe occurs at the end of (GO)(G = 2), the same strobes that were generated for G > 3 are produced. The only exception is that K_{72} is set or reset depending on the state of QSF. After this strobe, N will contain the quotient and A the remainder.

(GO) (G = 1)

During (GO) (G = 1), the final sign corrected quotient is produced in the S-adder and will be restored to the Q-register. The control enabled for this is $¢S_{0-35}/AQ_{36-71}$. Also, during this time the remainder is transferred from the A-register to N as a function of the Sign Indicator which may have been set during GS if the original dividend was negative. If the Sign Indicator is set, the control points $¢AQ_{0-35}/ZAQ_{0-35}$ $¢AQ_{36-71}/ZAQ_{36-71}$, $¢\overline{ZAQ}_{0-35}/N_{0-35}$, and $¢\overline{ZAQ}_{36-71}/N_{36-71}$ are enabled to transfer the one's complement of the remainder into N; control is established to set K_{72}. If the Sign Indicator is reset, control points $¢AQ_{0-35}/ZAQ_{0-35}$, $¢AQ_{36-71}/ZAQ_{36-71}$, $¢ZAQ_{0-35}/N_{0-35}$, and $¢ZAQ_{36-71}/N_{36-71}$ are enabled to transfer AQ straight through to N; K_{72} is reset for this case.

12.3.6 - contd

When the \$ strobe occurs at the end of (GO) (G = 1), the same strobes that were generated for G > 3 are produced except that $\$AQ_{36-71}$ is generated instead of $\$AQ_{0-35}$. Other differences are:

1) The K_{72} adder control flip-flop is set if the Sign Indicator is set; otherwise, K_{72} is reset.

2) The Sign Indicator is set if bit zero of the adder output is a one; otherwise reset the Sign Indicator.

3) The Zero Indicator is set if the adder output is zero; otherwise, reset the Zero Indicator.

4) The GO cycle is reset and the GF cycle is set.

5) The \$ strobe re-enters the Timing Pulse Generator to produce another \$ in about 400 nanoseconds.

At the end of (GO) (G = 1) the quotient with the correct sign is contained in Q_{36-17}.

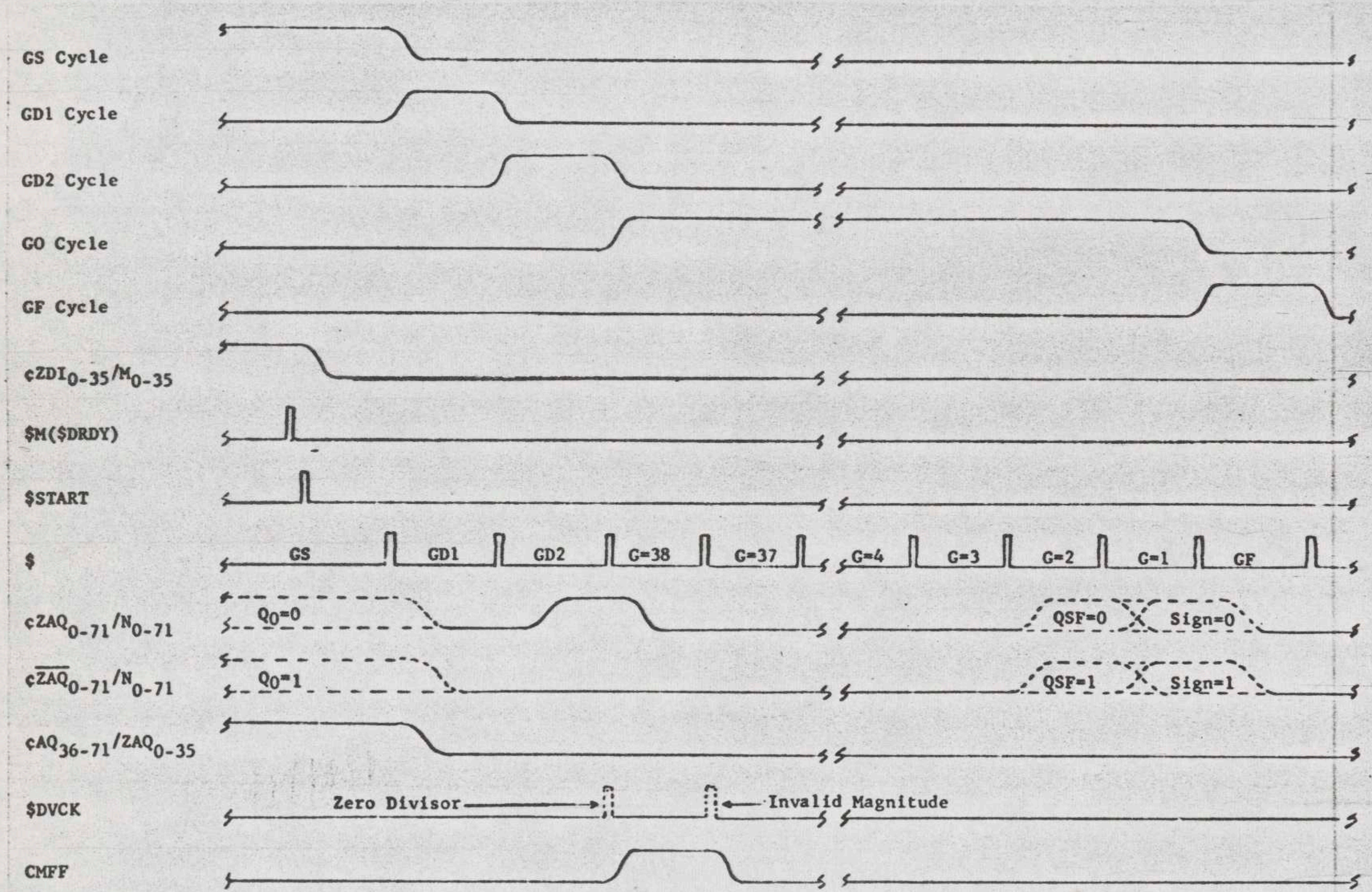
GF

During GF the sign correct remainder is produced in the S-adder and the control is established to transfer the remainder into A_{0-35}. The control enabled is ¢S_{0-35}/ AQ_{0-35}. When the \$ strobe occurs at the end of GF, the following events occur:

1) The GF cycle is reset.

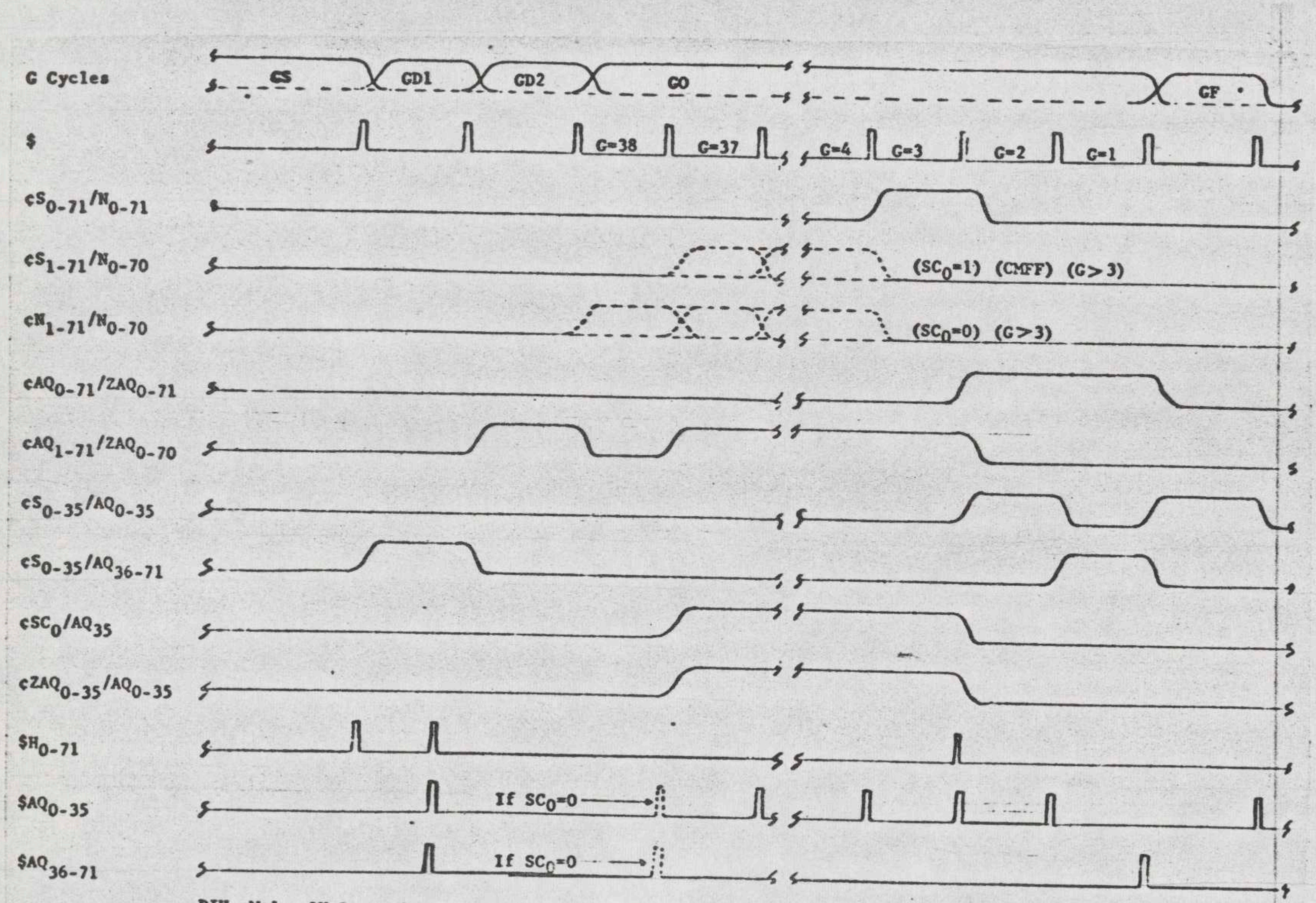
2) The $\$AQ_{0-35}$ is generated to load the remainder with its sign equal to that of the dividend into A.

3) It generates \$ LAST, which is sent to the CU to signal completion of the DIV instruction.


GS Cycle
GD1 Cycle
GD2 Cycle
GO Cycle
GF Cycle
$¢ZDI_{0-35}/M_{0-35}$
$M($DRDY)
$START
$
GS
GD1
GD2
G=38
G=37
G=4
G=3
G=2
G=1
GF
$¢ZAQ_{0-71}/N_{0-71}$
$Q_0=0$
QSF=0
Sign=0
$¢\overline{ZAQ}_{0-71}/N_{0-71}$
$Q_0=1$
QSF=1
Sign=1
$¢AQ_{36-71}/ZAQ_{0-35}$
$DVCK
Zero Divisor
Invalid Magnitude
CMFF

DIV -- Major OU Control Points and Strobes (Part 1)

Fig 12.3.6A FIG. 12.3.6a DIV Instruction, Timing Diagram
(Sheet 1 or 2)


G Cycles
CS
GD1
GD2
GO
GF
$
G=38
G=37
G=4
G=3
G=2
G=1
$¢S_{0-71}/N_{0-71}$
$¢S_{1-71}/N_{0-70}$
$(SC_0=1)$ (CMFF) $(G>3)$
$¢N_{1-71}/N_{0-70}$
$(SC_0=0)$ $(G>3)$
$¢AQ_{0-71}/ZAQ_{0-71}$
$¢AQ_{1-71}/ZAQ_{0-70}$
$¢S_{0-35}/AQ_{0-35}$
$¢S_{0-35}/AQ_{36-71}$
$¢SC_0/AQ_{35}$
$¢ZAQ_{0-35}/AQ_{0-35}$
$\$H_{0-71}$
$\$AQ_{0-35}$
If $SC_0=0$
$\$AQ_{36-71}$
If $SC_0=0$

DIV--Major OU Control Points and Strobes (Part 2)

FIG. 12.3.6a DIV Instruction, Timing Diagram
(Sheet 2 of 2)

Fig 12.3.6A

12.3.7 FIXED POINT SPECIAL INSTRUCTION GROUP

This group of nine instructions complete the fixed point repertoire. They are:

CWL, CMK, RPT, RPD, RPL, BCD, GTB, LREG, and SREG.

All of these instructions except CWL and CMK could be considered a subset of one of the six fixed point groups already discussed. CWL and CMK are unique in that they are the only fixed point instructions that use a GS - GO - GF sequence for their execution. The other seven instructions and the groups they are similar to are:

a) RPT, RPD, RPL - add group, single precision
b) BCD - divide group
c) GTB - shift group
d) LREG - add group, double precision
e) SREG - store group, double precision

Each of these instructions will be explained to point out the things that make them special.

12.3.7.1 RPT and RPL (Repeat and Repeat Link)

The RPT and RPL instructions are executed with the same sequence as those instructions in the single precision add group. The special feature of these two instructions is that a repeat mode flag is set during their execution which will enable the OU to keep track of the number of times the instruction to be repeated has been processed by the OU. The repeated instruction follows the RPT or RPL instruction. The repeat mode flag being set also allows the OU to detect a condition which will cause termination of the execution of the instruction being repeated. The termination may result from the instruction being repeated a specified number of times; or the state of the zero, sign, overflow, or carry indicator may cause a conditional terminate; or for RPL a Link Address = 0 will cause a terminate.

After the Repeat instruction is executed, the CU will continually send the OU the instruction to be repeated, with the correct operand, until a terminate condition occurs. After a termination condition has been detected, the OU will Abort. Abort is necessary because the instruction being repeated is overlapped and the OU will have started the execution of the repeated instruction one more time before terminate is recognized by the CU. Abort resets the OU to an initialized state to accept the next intended instruction.

12.3.7.1 - contd

The execution of the RPT and RPL is accomplished by transferring the operand from the ZDI bus into the M-register on $DRDY and from M to H on the first $ at the end of GS. During GF the operand is made available to the inputs of the X_0 index register via the S-adder. On the $ at the end of GF, and if bit 10 of H is a one, the operand is strobed into X_0. Bits 0-7 are the tally count and 11-17 are the terminate conditions. When $ LAST occurs for RPT or RPL, the Repeat/Repeat Link mode flip-flop (FTL) and the Sense Terminate flip-flop (STR) are set. These flip-flops set allow the repeated instruction to be processed until a terminate occurs. The tally is decremented by (GS) $(\overline{GF})$ ($) during each repetition of the repeated instruction. When the terminate occurs the terminate flag ¢TRMF to the CU goes high and this flag allows the next $ LAST to set the TRMF flip-flop in the CU. The $ that resets GF when ¢TRFM is high will set Abort in the OU.

When executing RPL, the CU does not permit overlap between an execution of the repeated instruction and the address modification for the next execution of the repeated instruction. Thus, if a Link Address = 0 condition occurs in the YS adder (bits 0-17) while the present repeated instruction is being processed, a terminate will be generated prior to executing the repeated instruction again using address zero for an operand fetch.

For the detailed control of these two instructions refer to the OU Specification Chart43D140232, page 11. A partial timing diagram for the RPT/RPL mode is shown on the following page.

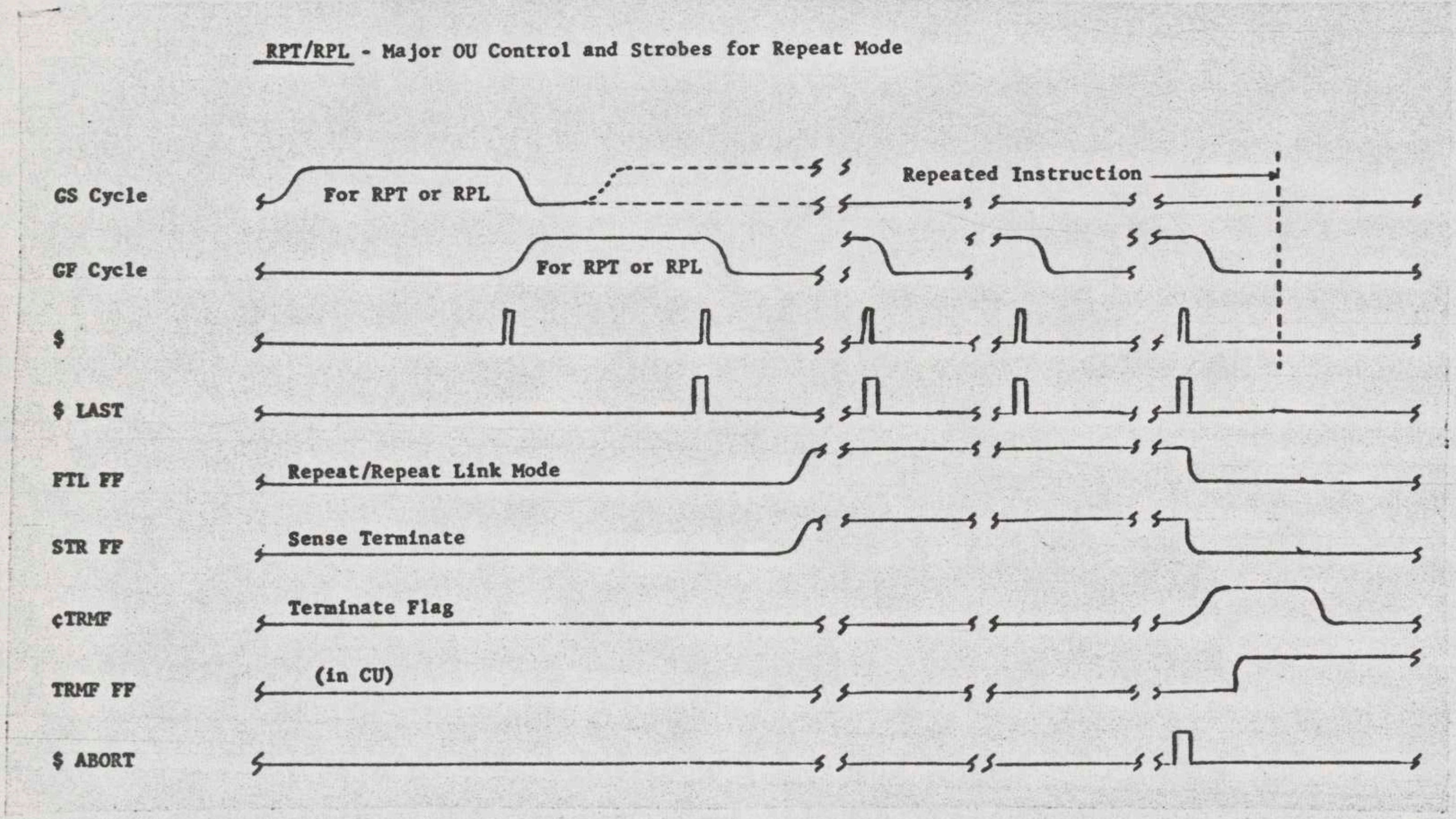

RPT/RPL - Major OU Control and Strobes for Repeat Mode
GS Cycle
For RPT or RPL
Repeated Instruction
GF Cycle
For RPT or RPL
$
$ LAST
FTL FF
Repeat/Repeat Link Mode
STR FF
Sense Terminate
¢TRMF
Terminate Flag
TRMF FF
(in CU)
$ ABORT

Fig 12.3.7.1

FIG. 12.3.7.1 RPT/RPL Instructions, Timing Diagram

12.3.7.2 RPD (Repeat Double)

When in the Repeat Double mode the repeated pair of instructions is always an even/odd pair and each instruction of the pair is executed the same number of times. The tally count is only decremented at the end of the GS cycle of the odd instruction, and a terminate condition is recognized only after completion of the odd instruction. As with RPT and RPL, the RPD is executed by transferring the operand from the ZDI bus into the M-register on $DRDY and from M to H on the first $ at the end of GS. During GF the operand is made available to the inputs of the X_0 index register via the S-adder. On the $ at the end of GF, and if bit 10 of H is a one, the operand is strobed into X_0. Bits 0-7 are the tally count and 11-17 are the terminate conditions. On $ LAST of the RPD, the Repeat Double mode flip-flop (FRD) is set and remains set until a terminate occurs. The Sense Terminate flip-flop (STR) will set on $ LAST of each even repeated instruction and reset on $ LAST of each odd. This forces the terminate flag ¢TRMF to the CU to come high only if a terminate condition exists during the odd repeated instruction. $ABORT is generated the same as for RPT and RPL.

For the detailed control of RPD, refer to the OU Specification Chart 43D140232, page 11. A partial timing diagram for RPD is shown on the following page.

RPD - Major OU Control and Strobes for Repeat Double Mode

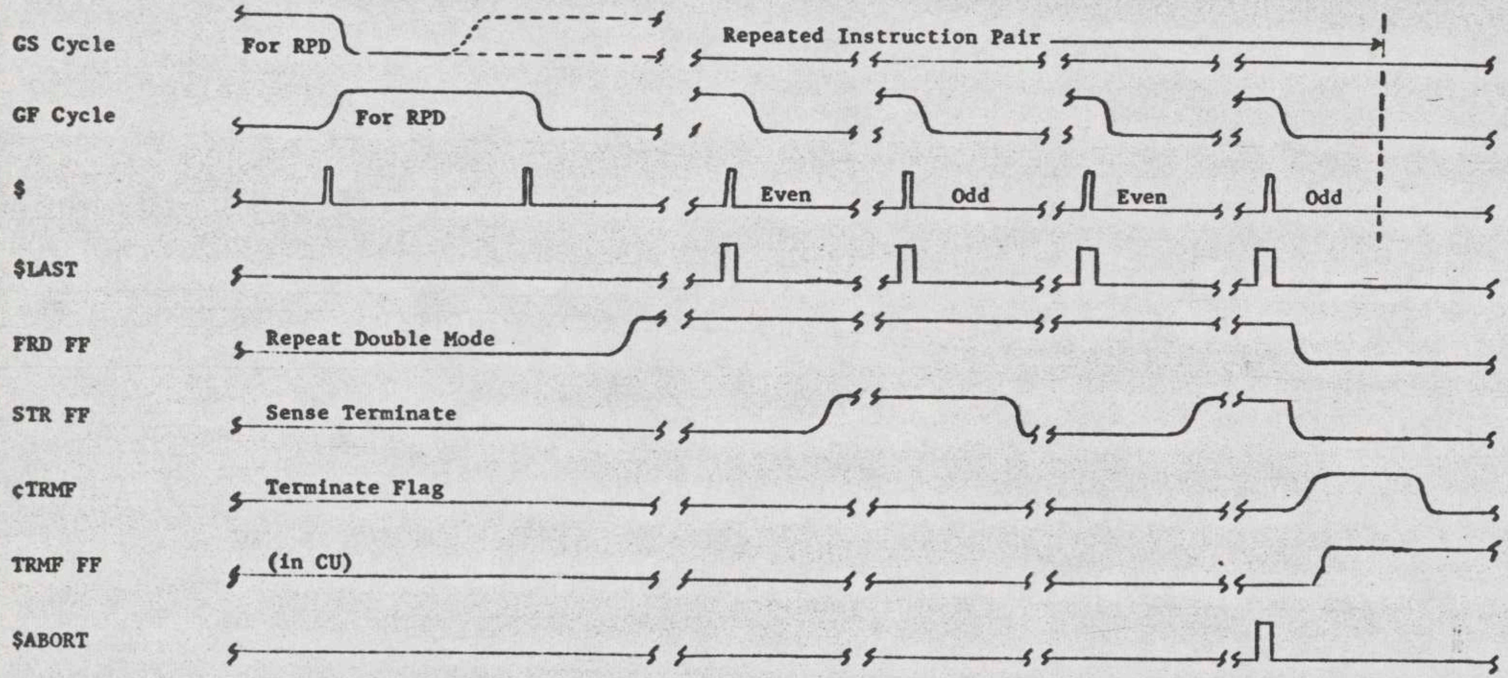

GS Cycle
For RPD
Repeated Instruction Pair
GF Cycle
For RPD
$
Even
Odd
Even
Odd
$LAST
FRD FF
Repeat Double Mode
STR FF
Sense Terminate
¢TRMF
Terminate Flag
TRMF FF
(in CU)
$ABORT

FIG. 12.3.7.2 RPD Instruction, Timing Diagram

12.3.7.3 BCD (Binary to Binary-Coded-Decimal)

The BCD algorithm used in this machine makes it possible to make the BCD execution a subset of the fixed point divide group. Each time the BCD instruction is executed, a 6-bit binary-coded-decimal digit is generated. The BCD number generated is in an (8, 4, 2, 1) code and the first two bits of the 6-bit number are always zero. Only valid BCD numbers $(000000)_2 = (0)_{10}$ through $(001001)_2 = (9)_{10}$ are generated.

The magnitude of the binary contents of the A-register are converted to a BCD digit by the use of a conversion constant from memory. Because of the BCD algorithm, if the binary number in A is a negative quantity, it is two's complemented and converted to the same positive number before the BCD conversion and the sign is retained in the sign indicator. After the BCD conversion the Q-register will hold the magnitude of the BCD sign. At the end of the execution of the BCD instruction, the A-register will contain the remainder of the original binary number with the original sign restored. If the BCD instruction is issued again, the remainder will now be the binary number used in the generation of the next BCD digit. Each time the BCD instruction is issued in the conversion of a binary number to a BCD digit, a new conversion constant is required from memory.

Because of the conversion constants necessary for the BCD conversion, the largest binary number that can be converted to BCD is $\pm (9,999,999,999)_{10}$. Although a binary number of -2^{35} or $-(34,359,738,368)_{10}$ can be contained in a 36-bit register, a conversion constant of 8×10^{10} which would be necessary to convert this number can not be represented with 36 bits. This implies that the 2^{34} bit should never be a one in the binary number that is to be converted.

12.3.7.3 - contd

Although the BCD conversion uses the divide hardware, the divide check fault will not set during BCD; that is, a divisor of zero (the conversion constant) or a magnitude check on the number to be converted is not performed.

GS

Execution of the BCD instruction is initiated by transferring the conversion constant on the ZDI bus into the M-register during GS when $DRDY is received from the CU. Also during GS the control is established to load the N-register with the contents of A (the number to be converted) in such a manner that a positive dividend will be guaranteed. The control enabled is ¢AQ_{0-35}/ZAQ_{0-35} and ¢ZAQ_{0-71}/N_{0-71} if the binary number is already positive, or ¢AQ_{0-35}/ZAQ_{0-35}, ¢$\overline{ZAQ}_{0-71}/N_{0-71}$, ¢ Set K_{72}, and ¢ Set Sign Indicator if the number is negative and needs to be two's complemented. On the $\underline{\$}$ strobe at the end of GS the N-register is loaded with A or $\bar{A}$, the H-register is cleared, K_{72} and the Sign Indicator will set if $A_0 = 1$, the QSF flip-flop and Zero Indicator will reset, and the GD1 cycle will be entered.

GD1

During GD1 control is established to load the S-adder output (positive dividend) into N, load a negative divisor (conversion constant) into H, and shift the AQ-register left one bit to form the first zero bit of the BCD digit. The controls enabled for these operations are ¢S_{0-71}/N_{0-71}, ¢AQ_{1-71}/ZAQ_{0-70}, ¢ZAQ_{0-71}/AQ_{0-71}, _¢ZAQ_{0-71}/AQ_{0-71}, ¢M_{0-35}/H_{0-35} if $M_0 = 1$, ¢$\overline{M}_{0-71}$ if $M_0 = 0$, and ¢ Set K_{72} if $M_0 = 0$ or reset if $M_0 = 1$. On the $ strobe at the end of GD1 the N-register is loaded

12.3.7.3 - contd

with the positive dividend, the H-register and K_{72} are strobed to make up the negative divisor, and the AQ-register is strobed for a left shift of one to force Q_{35} to zero. GD2 is set at this time.

GD2

During GD2 control is established to do another left shift of the AQ by one and load the G-counter with a count of 6. The control for the AQ left shift is the same as that used in GD1. When the strobe occurs at the end of GD2 and AQ is shifted to make Q_{34} and Q_{35} zero. The \$G strobe is produced to load a fixed count of 6 into the G-counter and the GO cycle is set.

(GO) (G > 3)

For the first three GO cycles or while G = 6, 5, and 4, three bits of the BCD digit are formed using the S-adder. Each carry out of bit zero of the adder becomes a bit of the BCD digit at the end of each GO cycle in the same manner that a quotient is generated for divide. At the end of each GO for G = 6, 5, and 4 and if $C_0 = 1$, the adder sum is shifted left one bit and loaded into N to form a new remainder by ¢S_{1-71}/N_{0-70}; or if $C_0 = 0$ the N-register itself is shifted left one bit to form a new remainder by ¢N_{1-34}/N_{0-34}, ¢N_{36}/N_{35}, and ¢N_{37-71}/N_{36-70}. For each GO the AQ is shifted left one bit the same as above and the new C_0 is inserted into Q_{35} by ¢C_0/AQ_{71} to form a new bit of the BCD inserted into Q_{35} by ¢C_0/AQ_{71} to form a new bit of the BCD inserted digit. The G count is decremented at the end of each GO cycle.

12.3.7.3 - contd

(GO) (G = 3)

During (GO) (G = 2), the last or fourth BCD digit is generated in the adder. This cycle is also similar to divide in that if $C_0 = 1$ the adder output is transferred into N with no shift by ¢S_{0-71}/N_{0-71}; or if $C_0 = 0$ the contents of N are unaltered. Again on the $ at the end of GO, the AQ is shifted left one bit as above and the new C_0 is inserted into Q_{35}. The H-register is cleared to zero and the G count decremented to 2. At this time the 6-bit decimal digit is contained in Q30-35 and a positive remainder is in N.

(GO) (G > 2)

During (GO) (G = 2), a set of redundant transfers take place to make the BCD instruction compatible with divide. The BCD digit in Q is transferred into the upper end of N, and the binary positive remainder in N is transferred into A via the adder. These transfers take place on $ at the end of GO and the G count is decremented to one.

(GO) (G = 1)

During (GO) (G = 1), the BCD character in N is returned to Q via the adder. The remainder in A is placed back in N in such a fashion to allow the remainder to be corrected in sign if necessary during GF. If the original binary number in A was positive, no sign correction is necessary so ¢AQ_{0-35}/ZAQ_{0-35} and ¢ZAQ_{0-71}/N_{0-71} are enabled to place the remainder as is into N. If the original number was negative, ¢AQ_{0-35}/ZAQ_{0-35},

12.3.7.3 - contd

¢$\overline{ZAQ}_{0-71}/N_{0-71}$, and ¢ Set K_{72} are enabled. The complete BCD character in N is restored to Q via ¢S_{0-35}/AQ_{36-71}. On the \$ strobe at the end of (G0) (G = 1), the G-counter is decremented to zero, the BCD character is loaded in Q, the remainder is loaded in N, K_{72} is set if required, and the GF cycle is set.

GF

During GF the remainder with correct sign is produced in the adder and ¢S_{0-35}/AQ_{0-35} is enabled to restore the remainder to A. On the \$ strobe at the end of GF the remainder is restored to A and K_{72} is unconditionally reset. The Zero Indicator is set if the adder output is zero which means that there is no remainder and the total binary to BCD conversion is complete.

For the detailed control of BCD, refer to the OU Specification Chart 43D140232, page 9.

12.3.7.4 GTB (Gray Code to Binary)

The GTB algorithm used in this machine made it possible to make the GTB instruction a subset of the Shifted Group. The major differences between the shifts and GTB are that the GTB always has 36 shifts, there can be no overshift and therefore no GN cycle. GTB is comparable to the Rotate A instruction with special bit control on bit 35. Also, during GTB the G-counter is used as a full 8-bit decrement counter. The GTB conversion is defined by the following algorithm, where R_i and S_i denote the contents of the i^{th} bit of the number before and after conversion:

$$S_0 = R_0$$

$$S_i = (R_i \cdot \overline{S_{i-1}}) \oplus (\overline{R_i} \cdot S_{i-1})$$

where $\oplus$ = exclusive "OR"

This GTB instruction will give meaningful results only if the gray coded number to be converted is of the cyclic (gray or reflected) code where only one bit of the coded number changes for ajdacent numbers. Specifically, the numbers are:

Gray Code	Binary	Decimal
0000	0000	0
0001	0001	1
0011	0010	2
0010	0011	3
0110	0100	4
0111	0101	5
0101	0110	6
0100	0111	7
1100	1000	8
1101	1001	9
1111	1010	10
etc.	etc.	etc.

12.3.7.4 - contd

The GTB conversion is accomplished with an exclusive OR, and the shift capabilities of the N-register. The gray coded number originally in the A-register is transferred into N for the conversion. The first shift is a simple rotation to preserve the sign bit. The next 35 left shifts are used to accumulate the correct binary bits which are inserted into N_{35} at the end of each GO cycle. Each bit generated is determined by the exclusive OR of bits N_0 and N_{35} that exist after the previous shift.

GS

This instruction is executed by transferring the gray coded number in the A-register into N by enabling ¢AQ_{0-35}/ZAQ_{0-35} and ¢ZAQ_{0-35}/N_{0-35} during GS. Also, during GS the control is established to load the G-counter with a count of 36. On the $ strobe at the end of GS, the N-register and G-counter are loaded, the H-register is cleared to zero, and the Zero and Sign Indicators are reset. The GS cycle is reset and GO is set.

(GO) (G = 36)

During (GO) (G = 36), control is established to left rotate N one bit and decrement the G-count by one. The control points enabled are ¢N_{1-35}/N_{0-34}, ¢N_0/N_{35}, and ¢FCD. On the $ strobe, the sign bit is rotated and the G-count becomes 35.

(GO) (G < 36)

During (GO) (G = 35), control points are enabled to left shift N by one bit and load the first generated binary bit into N_{35}. These controls are ¢N_{1-35}/N_{0-34} and ¢$N_0 \oplus N_{35}/N_{35}$. On the $ strobe at the end of (GO) (G = 35), the binary bit is loaded in N, the G-count is decremented, and a new GO cycle is entered. This process is repeated 34 times until the complete binary word is formed and the GF cycle is set.

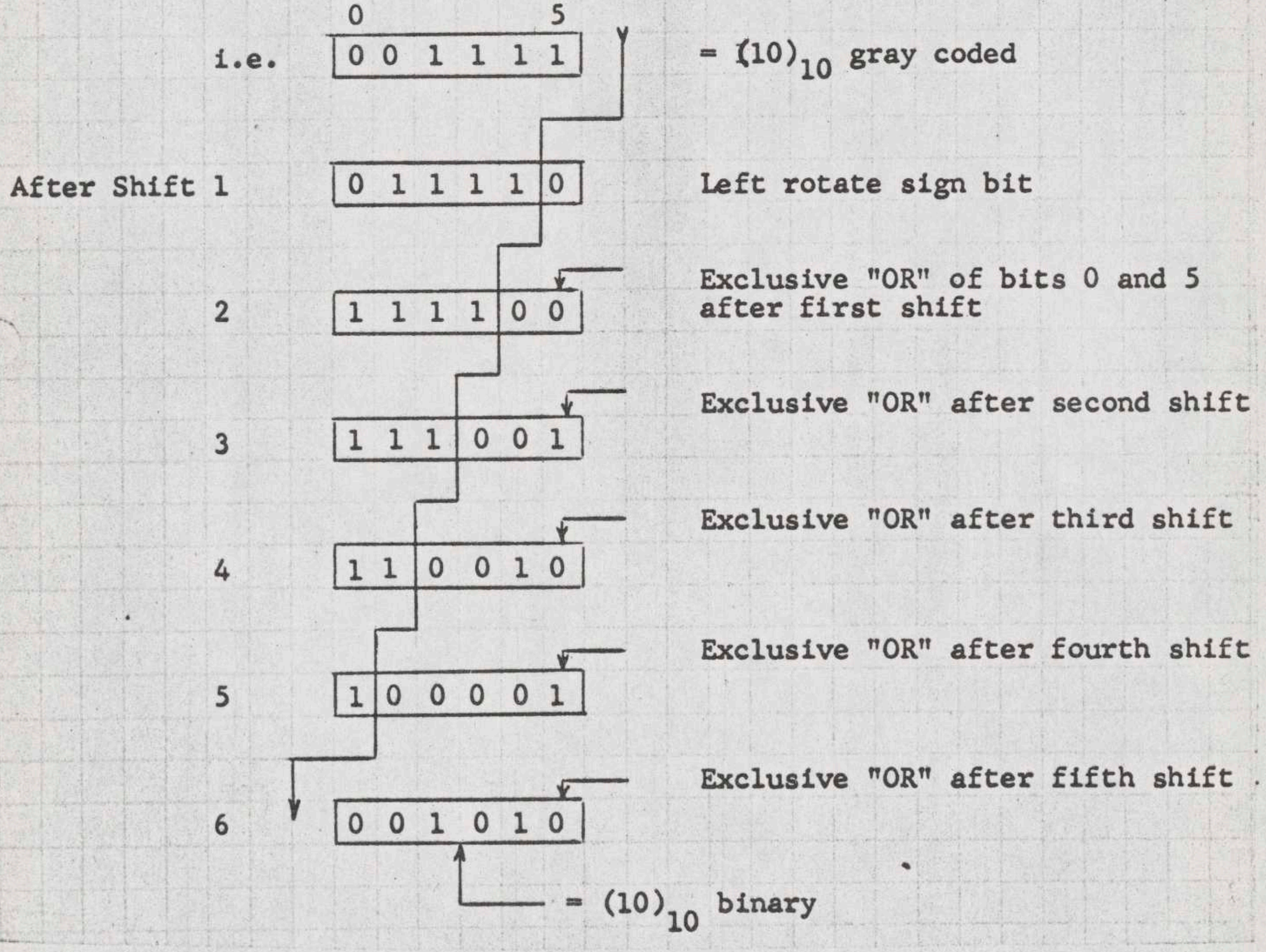
12.3.7.4 - contd

GF

During GF the binary word in N is restored to A via the S-adder. The control point enabled is ¢S_{0-35}/AQ_{0-35}. On the $ strobe at the end of GF the Sign Indicator will set if bit one of the adder is a one, and the Zero Indicator will set if the adder output is zero.

For the detailed control of GTB, refer to the OU Specification Chart, 43D140232, page 7. An example of GTB is shown on the following page.

EXAMPLE: Assume a 6-bit register holding a $(+10)_{10}$ gray coded number.


0
5
i.e.
0 0 1 1 1 1
= $(10)_{10}$ gray coded
After Shift 1
0 1 1 1 1 0
Left rotate sign bit
2
1 1 1 1 0 0
Exclusive "OR" of bits 0 and 5 after first shift
3
1 1 1 0 0 1
Exclusive "OR" after second shift
4
1 1 0 0 1 0
Exclusive "OR" after third shift
5
1 0 0 0 0 1
Exclusive "OR" after fourth shift
6
0 0 1 0 1 0
Exclusive "OR" after fifth shift
= $(10)_{10}$ binary

Tk 12.3.7.4

FIG. 12.3.7.4 GTB Instruction Example

12.3.7.5 LREG (Load Registers)

This instruction is executed with the same sequence of G cycles as those instructions in the double precision portion of the ADD Group. The special feature of LREG is that four consecutive double precision loads take place with only one operation code. In this respect LREG looks like the instruction referred to by a Repeat Instruction, where the repeat is to be done four times. That is, the CU will send the LREG instruction to the OU four times, followed each time by the appropriate double precision operands. Not until the OU is processing the fourth LREG instruction will the CU send a new instruction to the OU.

The AQ-register, E-register, Timer Register, and Index Registers 0-7 will be loaded with one instruction. The complete operation is done under control of the GRU flip-flops*. The GRU flip-flops are used to determine the order of loading the registers as follows:

GRU 3, 2, 1

000 = Load X_0, X_1, X_2, and X_3 registers

100 = Load X_4, X_5, X_6, and X_7 registers

010 = Load AQ-register

001 = Load E Register

The Index Registers are each 18 bits in length. Therefore, it is possible to load all index registers with two double precision operations. The load AQ portion of this instruction will be done much the same as (LDAQ), and is completely loaded in one double precision operation.

*See paragraph 12.8.12 for a description of the GRU flip-flops.

12.3.7.5 contd

The E-register is only 8 bits in length. However, it is loaded as a double precision operation. The E-register data is left justified in the operand word but is loaded into M_{64-71} during the first half of the double precision operation.

When the LREG decode comes true for the first operation, the control points ¢ZDI_{0-35}/M_{0-35} become enabled. On the first \$DRDY strobe for this double precision load, $\$M_{0-71}$ is generated to load the first half of the memory operand into the M-register. This \$DRDY strobe resets the DPFF, and is turned around as \$YRY and returned to the CU to signal that the next double precision pair may be prepared. This sequence is repeated for the remaining three double precision pairs. Upon entering the GS cycle for each pair the ¢M_{0-71}/H_{0-71} becomes enabled.

GRU = 000

When the \$ START pulse propagates the medium delay elements, the first operate strobe \$ occurs. This allows the following events:

1) The GS cycle to reset and the GF cycle to set.

2) \$H loads the first double precision operand into H.

3) The N-register is cleared to zero by \$N.

4) The ORB flip-flop is reset.

5) The \$ strobe re-enters the Timing Pulse Generator delay line network to produce another \$ in approximately 400 nanoseconds.

12.3.7.5 - contd

During this first GF cycle, the memory operand is made available on the S bus at the adder output. The GRU flip-flops being reset enable control lines for loading $X_0 - X_3$. These lines are:

a - For X_0, ¢S_{0-17}/ZRX_{0-17}

b - For X_{1-3}, ¢S_{0-71}/IBN

During GF, the GS cycle for the second DP pair will set allowing maximum overlap of the processing of the LREG instruction. When the \$ strobe occurs at the end of GF, the following events take place:

1) The GF cycle is reset.

2) GRU_3 is set.

3) $\$X_0 - \$X_3$ are generated to load the four upper index registers.

GRU = 100

During (LREG) (GRU's = 100), the same process as above is repeated except that during GF control is enabled to load $X_4 - X_7$. This control is ¢S_{0-71}/IBN. On the \$ strobe at the end of the second GF, GF is reset, GRU_3 is reset and GRU_2 set, and $\$X_4 - \$X_7$ are generated to load the four lower index registers.

12.3.7.5 - contd

GRU = 010

During (LREG) (GRU's = 010), the same process is again repeated except that during GF control points ¢S_{0-71}/AQ_{0-71} are enabled to load the full AQ-register. On the \$ strobe at the end of GF, GF is reset, GRU_2 is reset and GRU_1 set, and \$$AQ_{0-71}$ is generated to load the AQ-register.

GRU = 001

During (GS) (LREG) (GRU's = 001), the same process of accumulating the double precision word in the M-register is again repeated. The first half of the double precision load is done by enabling ¢ZDI_{0-7}/M_{64-71} for what will be the E-register contents. For the second half of the double precision load, the control ¢ZDI_{0-35}/M_{0-35} is enabled. After the double precision operand is accumulated during (GRU's = 001), the CF may then send a new operation code to the OU to allow the normal overlap of instruction execution.

At the same time that the M-register is being loaded, control is enabled to transfer the E-register operand from M to D. This control point is ¢M_{64-71}/D_{0-7}.

When the \$ strobe occurs at the end of GS, the following events take place:

1) GS is reset and GF set.

2) \$D is generated to load D.

During the final GF of LREG, ¢D_{0-7}/E_{0-7} are enabled to load the last operand. The \$ strobe at the end of GF will load the E-register and GF and GRU_1 will reset.

The \$ will become \$ LAST to complete the LREG instruction.

12.3.7.6 SREG (Store Registers)

This instruction is executed with the same sequence of G cycles as those instructions in the double precision portion of the Store Group. The special feature of SREG is that four consecutive double precision stores take place with only one operation code. In this respect, SREG looks like the instruction referred to by a Repeat Instruction, where the repeat is to be done four times. That is, the CU will send the SREG instruction to the OU four times, followed each time by the appropriate transfer from the OU of the double precision store data. Not until the OU is processing the fourth SREG instruction will the CU be able to send a new instruction to the OU.

The AQ-register, E-register, Timer Register, and Index Registers 0-7 will be stored with one instruction. The complete operation is done under control of the GRU flip-flops*. The GRU flip-flops are used to determine the order of storing the registers as follows:

GRU 3, 2, 1

000 = store X_0, X_1, X_2 and X_3 registers

100 = store X_4, X_5, X_6 and X_7 registers

010 = store AQ-register

001 = store E and Timer Registers

*See paragraph 12.8.12 for a description of the GRU flip-flops.

12.3.7.6 - contd

The index registers are each 18 bits in length. Therefore, it is possible to store all index registers with two double precision operations. The store AQ Portion of this instruction will be done much the same as (STAQ), and is accomplished in one double precision operation. The E-register and the Timer are stored during the last double precision operation. The exponent register is left justified in the first word and the Timer is left justified in the second word of the last double precision operation.

When the SREG decode has settled for the first time and the GS cycle is set, the GC cycle flip-flop sets. This allows the ZAQ and ZDO switches to be enabled by ¢X_0/ZAQ_{0-17}, ¢Z_1/ZAQ_{18-35} and ¢ZAQ_{0-35}/DO_{0-35}. These switches enable The X_0 and X_1 Index Registers to be present at the DO bus.

When the $ set GS pulse was generated for this instruction, it was delayed by about 40 nanoseconds to become $YRY and it becomes $SDY.

$SDY is sent to the CU to initiate the memory store cycle prior to the occurrence of the first $ strobe in the OU.

GRU = 000

When the $ START pulse propagates the medium delay elements, the operate strobe $ occurs. This allows the following events:

1) The GS cycle is reset and the GT cycle is set.

2) The O-register busy flip-flop ORB is reset.

3) A level consisting of DP Stores and GT goes high.

4) The $ strobe re-enters the Timing Pulse Generator, but because a store is being executed, this will not produce a subsequent $ strobe.

12.3.7.6 - contd

Within a minimum time of 300 nanoseconds after $DSY, the first half of the store operation will be complete and the first $DS will be received from the Interface Frame to signal this fact. $DS will produce a $ t strobe which will:

1) Reset the GC and GT cycle flip-flops.

2) Set the GF and GFT cycle flip-flops.

3) $ will be "ANDed" with the DP Stores and GT level to produce $DP which is delayed by approximately 270 nanoseconds before being sent to the Interface Frame to signify that the second data word is ready to be stored.

When the GF cycle was set, the ZAQ and ZDO switches are enabled by control points ¢X_2/ZAQ_{0-17}, ¢X_3/ZAQ_{18-35} and ¢ZAQ_{0-35}/DO_{0-35}. This allows the X_2 and X_3 Registers to be present at the DO bus before the OU sends the $DP pulse to the Interface Frame for the second half store.

The $DP strobes the data into the Data Out Buffer bits 36-71 and is returned as the second $DS indicating that the second half data has been received. This $DS will generate a $ pulse which will:

1) Reset the GF and GFT cycles.

2) GRU_3 is set in preparation for the next double word transfer.

12.3.7.6 - contd

GRU = 100

The CF will have sent the SREG instruction to the OU again by the end of the first GS cycle, so the OU is ready to go through another double word transfer. This will allow the SREG to overlap itself in the OU. The same execution procedure as above is again repeated except that for (SREG) (GRU's = 100), index registers X4 - X7 are stored into memory. At the end of the second GF cycle GRU_3 is reset and GRU_2 is set.

GRU = 010

During (SREG) (GRU's = 010), the same double precision store sequence as above is executed. This time control points are enabled to store the A- and the Q-registers via the ZAQ and ZDO switches during the first and second half of the instruction. At the end of the third GF, GRU_2 is reset and GRU_1 is set. The CF cannot send the next sequential instruction to the OU unitl the GRU count is 001.

GRU = 001

During the final store operations which take place for (SREG) (GRU's = 001), the E-register and Timer are stored. The E-register is stored first via $¢E/DO_{0-7}$. The Timer is stored via $¢TR/ZAQ_{0-23}$ and $¢ZAQ_{0-35}/DO_{0-35}$. When the $ strobe occurs at the end of the fourth GF cycle, GRU_1 and GF are reset. This $ strobe becomes $ LAST which is sent to the CF to signal completion of the SREG instruction.

For detailed control of SREG, refer to the OU Specification Chart 43D140232, page .

12.3.7.7 CWL (Compare with Limits)

The CWL instruction is a special compare to determine the algebraic magnitude of a memory word relative to the contents of the A- and the Q-registers. The comparison will determine whether an algebraic number from memory is contained within a closed interval bounded by the algebraic contents of A on one end and the algebraic contents of Q on the other. It makes no difference if the algebraic magnitude of A is less than, equal to, or greater than Q.

that is, $C(A) \leq c(Y) \leq c(Q)$

Example $(-2) \leq (+5) \leq (+7)$ (within limits)

or $c(A) \geq c(Y) \geq c(Q)$

Example $(+7) \geq (+5) \geq (-2)$ (within limits)

If a comparison is made and the number is within the limits of A and Q, the Zero Indicator will be set. If the number is not within the limits of A and Q, the Zero Indicator will not set. The state of the Sign and Carry indicators will reflect the magnitude relationship between the C(Y) and C(Q), and the signs C(Y) and C(Q) will be known. That is,

Sign	Carry	Relation between C(Q) and C(Y)	Signs of C(Q) and C(Y)
0	0	$C(Q) > C(Y)$	$C(Q)_0 = 0$, $C(Y)_0 = 1$
1	0	$C(Q) \geq C(Y)$	
1	1	$C(Q) < C(Y)$	$C(Q)_0 = C(Y)_0$

12.3.7.7 - contd

CWL is executed with a GS-GO-GF sequence in the OU. When the CWL decode comes true, the control point $¢ZDI_{0-35}/M_{0-35}$ on the IM switch are enabled. When the $\$M_{0-71}$ is generated to load the C(Y) into the M-register.

$ START generation is identical to any other instruction.

GS

During GS, the IH, ZAQ, and IN switches are enabled to load the operand registers H and N for the first comparison. IH is enabled by $¢M_{0-71}/H_{0-71}$ so that the one's complement of C(Y) will be loaded into H for use in the comparison (subtraction) with C(A) and C(Q). The ZAQ and IN switches are enabled by $¢AQ_{0-35}/ZAQ_{0-35}$ and $¢ZAQ_{0-35}/N_{0-35}$ in order to load the C(A) into N.

When the $ START pulse propagates the medium delay elements, the first operate $ occurs. This allows the following events:

1) The GS cycle is reset and the GO cycle is set.

2) $\$N_{0-71}$ and $\$H_{0-71}$ are generated to load H and N.

3) The K_{72} adder control flip-flop is set to complete the two's complement of the C(Y).

4) The Zero and Sign Indicators are reset.

5) The $ strobe re-enters the Timing Generator to produce another strobe in about 400 nanoseconds.

12.3.7.7 - contd

GO

During GO the C(Y) are subtracted from the C(A) in the S-adder. Controls to set the Sign Indicator may come high depending on the subtraction results. Also, during GO, control points ¢AQ36-71/ZAQ0-35 and ¢ZAQ$_0$-35/N$_0$-35 are established to load the C(Q) into N at the end of GO. When the $ strobe occurs at the end of GO, the following events take place:

1) The GO cycle is reset and the GF cycle is set.

2) $\$N_{0}$-71 is generated to load the C(Q) into N.

3) The Sign Indicator is set if

$$(\overline{S_0})\ [SC_0\overline{SC_1} + \overline{SC_0}SC_1] + (S_0)\ [SC_0SC_1 + \overline{SC_0}\,\overline{SC_1}].$$

4) The $ re-enters the Timing Generator to produce another strobe in about 400 nanoseconds.

GF

During GF the C(Y) are subtracted from the C(A) in the S-adder. Controls may come high to set the sign, zero, and/or carry indicators depending on the subtraction results. When the $ strobe occurs at the end of GF, the following events occur:

1) The GF cycle is reset.

2) The K_{72} flip-flop is reset.

3) The Sign Indicator will set for the same conditions as above.

12.3.7.7 - contd

4) The Carry Indicator will set if SC_0 (a carry from bit zero of the adder) is a one.

5) The Zero Indicator will set if the Sign Indicator was set at the end of GO and it will not be set at the end of GF, OR if the Sign Indicator was not set at the end of GO and it will be set at the end of GF.

That is, [(Sign Ind) (¢ $\overline{\text{Set Sign Ind}}$)] +

[($\overline{\text{Sign Ind}}$) (¢Set Sign Ind)]

6) This $ strobe becomes $ LAST which is sent to the CU to indicate completion of the CWL instruction.

12.3.7.7 - contd

Example 1:

Compare C(Y) = (Zero)

with C(A) = (+1) and C(Q) = (-1)

A = 0001
Q = 1111 (2's complement)
Y = (1's complement in H-register)

GO

$$\begin{array}{lr} & 1 \\ A & 0001 \\ \underline{-Y} & \underline{1111} \\ C & 0001 \end{array} \qquad K_{72} \text{ set}$$

Sign indicator not set.

GF

$$\begin{array}{lr} & 1 \\ Q & 1111 \\ \underline{-Y} & \underline{1111} \\ C & 1111 \end{array}$$

¢ Set Sign indicator is true
because of (S_0) (SC_0SC_1)

12.3.7.7 - contd

The Zero Indicator is set. [($\overline{\text{SIGN}}$)(¢Set Sign) = 1]

The Carry Indicator is set. [$SC_0 = 1$]

The Sign Indicator is set. [(S_0) (SC_0SC_1)= 1]

The final indicator conditions say:

1) The C(Y) is within the limits of C(A) and C(Q).

2) The C(Q) is less than C(Y).

3) The sign of Q is negative and the sign of Y is positive.

12.3.7.8 CMK (Compare Masked)

The CMK instruction compares the C(Y) with C(A) on a bit-for-bit basis. However, any number of bits may be ignored in the comparison. The bits to be ignored are denoted by "ones" in the Mask (Q) register. An exclusive "OR" is performed between C(Y) and C(A) to determine if any bits differ. This operation is followed by a logical "AND" of the results of the "EX OR" and the mask word in Q-register. All bits with "ones" in Q will become "Zeros" in the result of the logical "AND" and comparison is made only on the bits which differed and are unmasked. The final results of CMK conditionally set the Zero and Sign indicators. In the general case the zero indicator will set if the exclusive "OR" of C (y) and C(A) "AND" C(Q) equal zero; that is, Set zero if

$$[C(Y) \oplus C(A)]\ [C(Q)] = 0$$

Two unique cases for which the Zero Indicator will set are:

1) Each bit of C(Y) = each bit of C(A).

2) C(Q) is all ones.

That is, Set zero if $[C(Y)_{0-35} = C(A)_{0-35}] + [C(Q)_{0-35} = 1-1]$

The Sign Indicator will be set only if bit zero of Y is not the same as bit zero of A "AND" bit zero of Q is a zero.

That is, Set sign if $(A_0 \neq Y_0)\ (Q_0 = 0)$

The CMK instruction is executed with a GS - GO--GF sequence as is CWL. The loading of C(Y) into M-register and $ START generation is per normal procedure.

12.3.8 Floating Point Add Group

This group of 17 instructions consists of floating point additions, subtractions, loads, and compares. The major data paths and control points for these instructions are shown in Fig. 12.3.8. In the execution of these instructions, adder registers H and N are loaded from some source, the exponent registers D and E are loaded from some source, the S-adder performs an operation determined by the adder control flip-flops, K_1, K_2, K_3, and K_{72}, the ES-adder performs the required operation, and the results of these operations are strobed into a designated register and/or examined for the purpose of setting Arithmetic Indicators. (For detailed description of the S-adder operation see 12.10.)

To illustrate the control used in the execution of these instructions, several instructions will be explained in detail. For the detailed control of all instructions in this group, refer to the OU Specification Charts 43D140232, page 6.

In the explanation of all the example instructions, the following conditions will be assumed:

1) The instruction has been strobed into the Q-register.

2) The GS cycle flip-flop has been set.

3) The DDF flip-flop has been set indicating the operation decode has settled.

NOTE: Reference to Fig. 12.3.8 and the partial timing diagram at the end of each example will aid in the understanding of the instruction execution and implementation.

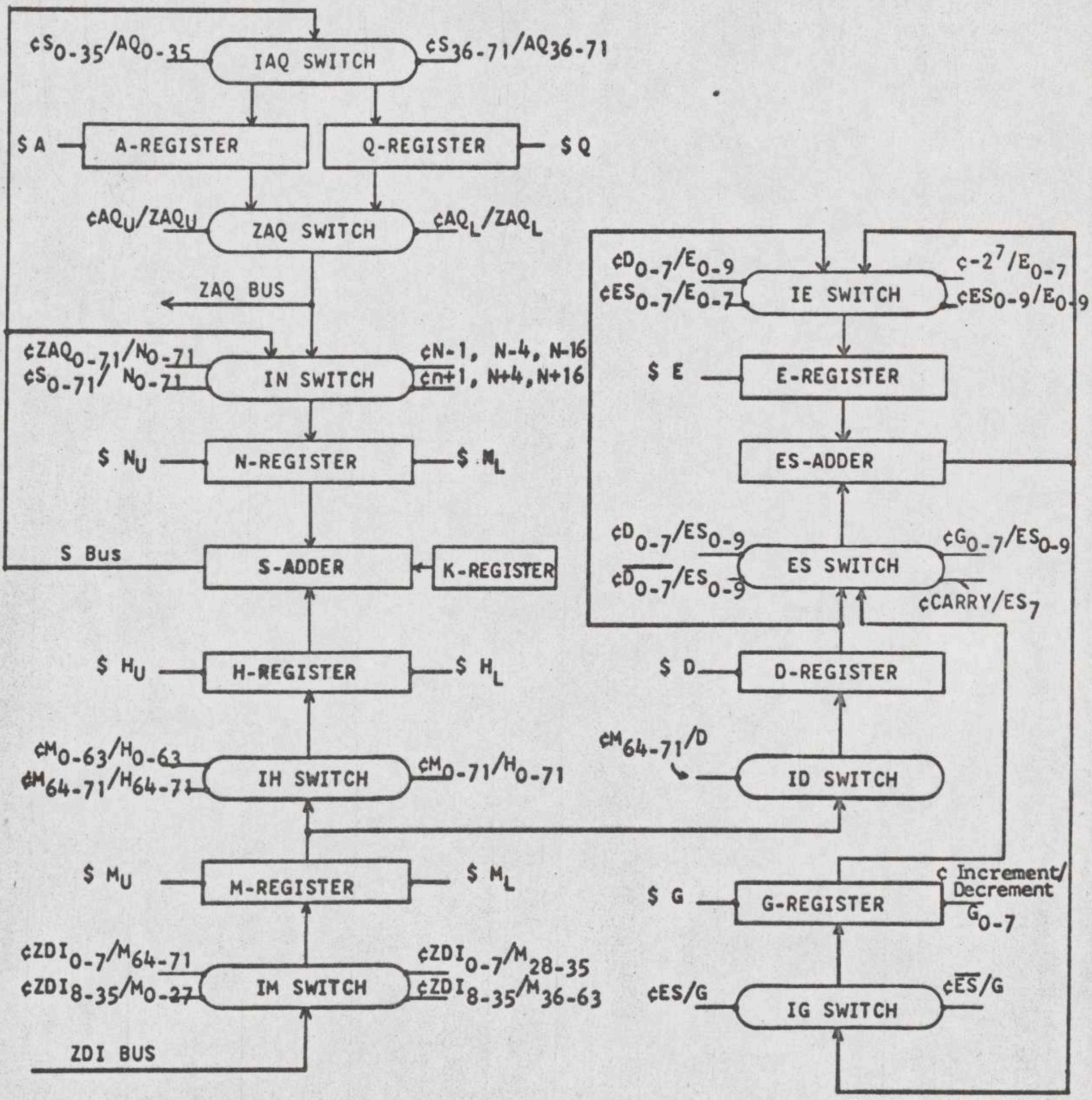

¢S0-35/AQ0-35
IAQ SWITCH
¢S36-71/AQ36-71
$ A
A-REGISTER
Q-REGISTER
$ Q
¢AQU/ZAQU
ZAQ SWITCH
¢AQL/ZAQL
ZAQ BUS
¢ZAQ0-71/N0-71
¢S0-71/N0-71
IN SWITCH
¢N-1, N-4, N-16
¢n+1, N+4, N+16
$ NU
N-REGISTER
$ NL
S Bus
S-ADDER
K-REGISTER
$ HU
H-REGISTER
$ HL
¢M0-63/H0-63
¢M64-71/H64-71
IH SWITCH
¢M0-71/H0-71
$ MU
M-REGISTER
$ ML
¢ZDI0-7/M64-71
¢ZDI8-35/M0-27
IM SWITCH
¢ZDI0-7/M28-35
¢ZDI8-35/M36-63
ZDI BUS
¢D0-7/E0-9
¢ES0-7/E0-7
IE SWITCH
¢-2^7/E0-7
¢ES0-9/E0-9
$ E
E-REGISTER
ES-ADDER
¢D0-7/ES0-9
¢D̄0-7/ES0-9
ES SWITCH
¢G0-7/ES0-9
¢CARRY/ES7
$ D
D-REGISTER
¢M64-71/D
ID SWITCH
$ G
G-REGISTER
¢ Increment/Decrement G0-7
¢ES/G
IG SWITCH
¢ĒS/G

FIG. 12.3.8

NOTES: (1) ¢ = Control
(2) $ = Strobe
(3) U = Upper
(4) L = Lower

FIG. 12.3.8 Floating Point Add Group, Major Flowchart

12.3.8.1 Single Precision

The instructions comprising the single precision floating point add group can be defined by two classifications:

1) Instructions requiring normalization (FAD, FSB).

2) Instructions that do not require normalization (ADE, FCMG, FCMP, FLD, FSZN, LDE, UFA, UFS).

Normalized

Under this heading fall the two instructions FAD and FSB.

Example 1: Add (FAD)

During this instruction a floating point number in a memory location Y is added to the contents of the Floating Register (EAQ). This is done by first comparing the exponents, aligning the mantissas, add the mantissas, normalize the result, and store the results in EAQ.

The presence of the FAD instruction code on the ZOR lines together with DPFF reset, BCFF reset, and DIC reset, enable the IM-switch. The control points ¢ZDI_{0-7}/M_{64-71}, and ¢ZDI_{8-35}/M_{0-27} are enabled presenting the operand lines (ZDI bus) to the proper M-register inputs. When the operand ready strobe ($ DRDY) is received from the C , the DPFF not being set allows the M-register strobe ($ M0-71) to be generated and strobe the FAD operand into the M-register. The $ DRDY pulse is turned around as $ YRY and returned to the CU to signal that a new instruction may be prepared. $ YRY sets the operand ready flip-flop (YRY) to allow a $ START to be generated by a last of three pulse detectors. $ START sets the OU Busy flip-flop (OUB), resets YRY and DDF, and transfers the operation code from the O-register to the C-register. It also enters the timing pulse generator delay line network.

GS

The IH-switch is enabled by ¢M_{0-63}/H_{0-63} and the ID-switch is enabled by ¢M_{64-71}/D. These enabled switches allow the mantissa from memory to be present at the inputs to the H-register and the exponent to be present at the inputs to the D-register.

When the $ START pulse propagates the medium delay elements, the operate strobe ($) occurs. This pulse causes the following events:

1) A $ H to strobe the mantissa from M into H.

2) A $ N to clear the N-register to zero.

3) A $ D to strobe the exponent from M into D.

4) ZERO and SIGN to reset.

5) GS to reset and GE to set.

6) $ enters the timing pulse generator delay line network to produce another $ pulse in approximately 400 nanoseconds.

GE

During the GE cycle the two exponents are compared by subtracting D from E, the difference if any is stored in G, the mantissa with the lesser exponent is stored in N, the other mantissa is stored in H, and the larger exponent is stored in E. The control lines that become enabled to accomplish this are:

1) ¢AQ_{0-71}/ZAQ_{0-71} unconditionally

2) ¢ZAQ_{0-71}/H_{0-71} unconditionally

12.3.8.1 - contd

3) ¢D/E	unconditionally
4) $¢\overline{D}/ES$	unconditionally
5) $¢CARRY/ES_7$	unconditionally
6) $¢ZAQ_{0-71}/N_{0-71}$	if $ES_9 + (S=0) + (ES=0)$
7) $¢S_{0-71}/N_{0-71}$	if $\overline{ES_9 + (S=0) + (ES=0)}$
8) ¢ES/G	if $(ES_0)\ [(ES \neq 0)(/ES/<72)(ES_9 + S \neq 0)]$
9) $¢\overline{ES}/G$	if $(\overline{ES_0})[(ES \neq 0)(/ES/<72)\ (ES_9 + S \neq 0)]$

At the end of 400 nanoseconds a \$ pulse will cause the following events:

1) A \$ H to strobe the mantissa from AQ into H if $(\overline{ES_9 + S = 0 + ES = 0})$.

2) A \$ N to strobe a mantissa into N if $(/ES/< 72) + (ES = 0) + (S=0)\ (ES_9)$.

3) A \$ E to strobe the exponent from D into E if ES_9.

4) A \$ G to strobe the proper count into G_{0-7} unconditionally.

5) GE to reset unconditionally.

6) GA to set if $(ES \neq 0)\ (/ES/<72)\ (ES_9 + S = 0)$.

12.3.8.1 - contd

7) GO to set if $(ES = 0) + (/ES/>71) + (\overline{ES_9})\ (S = 0)$.

8) $ enters the timing pulse generator delay line network to produce another $ strobe.

GA

During this cycle the mantissa in N is aligned so that its binary point falls at the same place as that of the mantissa in H. This is accomplished by shifting N right the number of places indicated by the G-counter. The control lines that become enabled to accomplish these events are:

1) ¢N-16 (¢N_{0-55}/N_{16-71}, ¢N_0/N_{0-15}) if $G > 14$.

2) ¢N -4 (¢N_{0-67}/N_{4-71}, ¢N_0/N_{0-3}) if $15 > G > 2$.

3) ¢N-1 (¢N_{0-70}/N_{1-71}, ¢N_0/N_0) if $3 > G > 0$.

4) ¢N+1 (¢N_{1-71}/N_{0-70}) if REV is set.

5) LSA enabled if $(G = 1) + (G = 4) + (G = 16) + (REV)$.

At the end of approximately 230 nanoseconds a $ strobe will cause the following events.

1) A $ N to strobe the shifted contents of N back into N.

2) A $ G to decrement G by 16 if ¢N-16, by 4 if ¢N-4, by 1 if ¢N-1.

3) REV to set if $(G = 15) + (G = 3)$.

4) GA to reset and GO to set if LSA enabled.

5) $ enters the timing pulse generator delay line network to produce another $ strobe.

GA will be repeated as many times as required to align the mantissa. The GA sequence will terminate on the first $ pulse following the enabling of LSA.

GO

During this cycle the contents of N and H are added in the S-adder, the sum is stored in N, and H is cleared to zero. The control line that becomes enabled is:

¢S_{0-71}/N_{0-71} to transfer the output of the S-adder to N.

At the end of 400 nanoseconds a $ pulse will cause the following events:

1) A $ H that clears H to zero.

2) A $ N to strobe the output of the S-adder into N.

3) REV to set if an overflow occurred.

4) SIGN to set if the sum is negative.

5) ZERO to set if the sum is zero.

6) CARRY to set if there is a carry out of bit position zero.

7) CARRY to reset if there is no carry out of bit position zero.

8) K_{72} to reset.

9) GO to reset.

10) GN to set if an overflow occurred or S_0 and S_1 are the same.

11) GF to set if no overflow occurred and S_0 and S_1 are different.

12) $ enters the timing pulse generator delay line network to produce another $ strobe.

GN

During this cycle the mantissa in N is normalized. If an overflow occurred the mantissa is shifted one place right and the G-counter is decremented by 1. In all other cases the mantissa is shifted left until N_0 and N_1 are not alike and the G-counter is incremented by 1 for each place that the mantissa is shifted. The control lines that become enabled to accomplish this are:

1) ¢N_{0-70}/N_{1-71}, complement N_0) if REV set.

2) ¢N+16 (¢N_{16-71}/N_{0-55}) if N_0 through N_{15} are all the same.

3) ¢N+4 (¢N_{4-71}/N_{0-67}) if $\overline{\text{¢N+16}}$ and N_0 through N_3 are all the same.

4) ¢N +1 (¢N_{1-71}/N_{0-70}) if $(\overline{\text{¢N+16}})(\overline{\text{¢N+4}})$ and N_0 and N_1 are the same.

5) LSN enabled if (¢N+16) $(N_{15} = N_{16} \neq N_{17})$ + (¢N+4) $(N_3 = N_4 \neq N_5)$ + (¢N+1) $(N_0 = N_1 \neq N_2)$ + (REV).

At the end of 230 nanoseconds a $ pulse will cause the following events:

1) A $ N to strobe the shifted contents of N back into N.

2) A $ G to increment the G-counter by 16 if ¢N+16, by 4 if ¢N+4, by 1 if ¢N+1, or decrement G by 1 if REV is set.

3) REV to set if (¢N+16) $(N_{15} \neq N_{16})$ + (¢N+4) $(N_3 \neq N_4)$

4) REV to reset if REV is set.

5) GN to reset and GF to set if LSN enabled.

6) $ enters the timing pulse generator delay line network to produce another $ strobe.

GN will be repeated as many times as required to normalize the mantissa and will terminate on the first $ pulse following the enabling of LSN.

GF

During this cycle the mantissa in N is transferred to AQ, the count in G is subtracted from the exponent in E, and EOFL and EUFL are set if required. If the mantissa is zero, the exponent will be forced to -128. The control lines that become enabled to accomplish this are:

1) ¢S_{0-71}/AQ_{0-71}.

2) ¢$\overline{G}$/ES and ¢ CARRY/ES_7.

3) ¢ES/E if mantissa $\neq$ zero.

4) ¢ - 2^7/E if mantissa = zero.

At the end of 400 nanoseconds a $ pulse will cause the following events:

1) A $ AQ to strobe the mantissa into AQ.

2) A $ E to strobe the exponent into E.

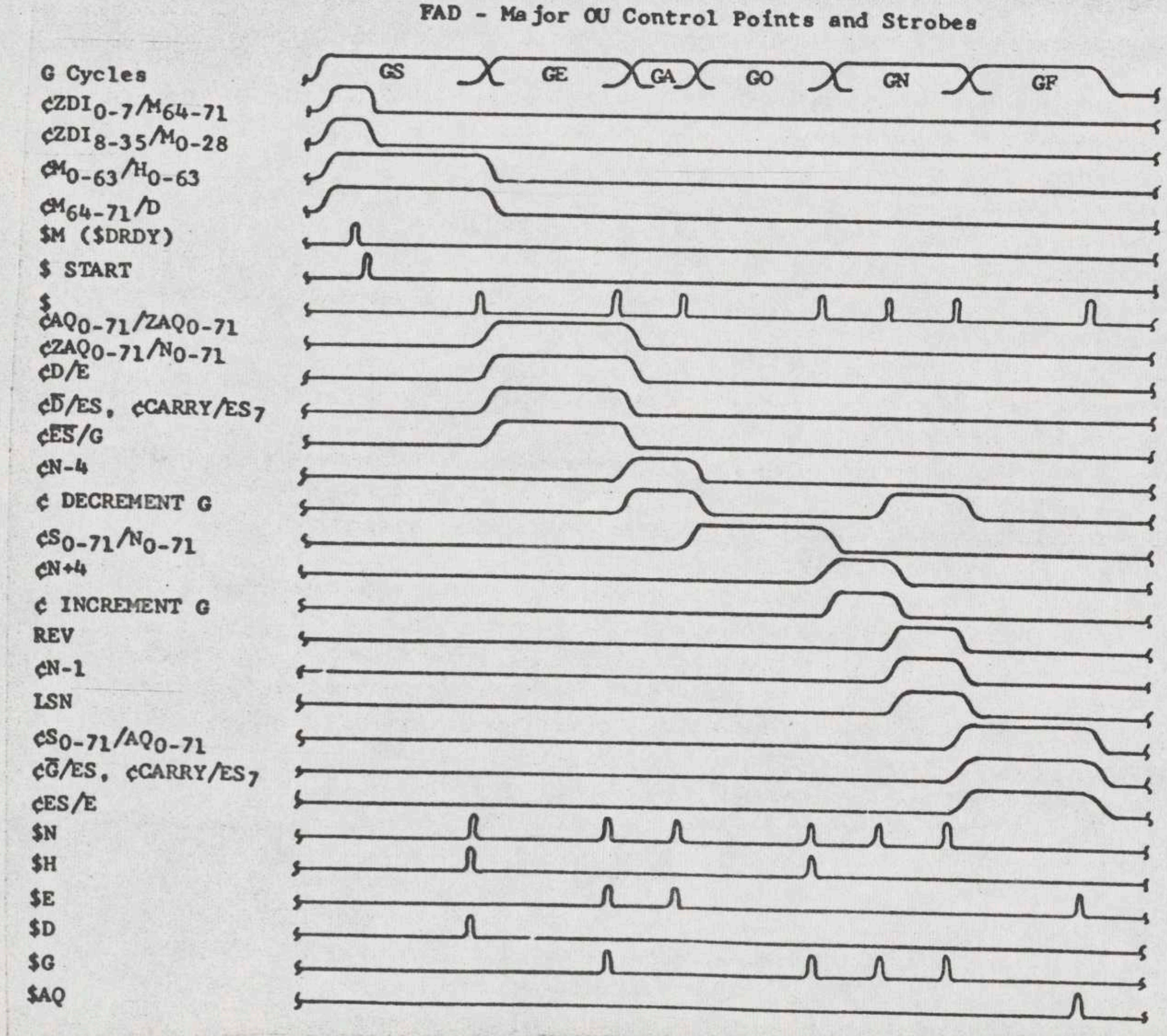
3) EOFL to set if the subtraction of G from E caused the exponent to exceed +127.

4) EUFL to set if the subtraction of G from E caused the exponent to become smaller than -128.

5) GF to reset.

6) OUBF to reset if no other instruction has been initiated.

7) $ will generate a $ LAST, which is sent to the CU to indicate completion of the FAD instruction.


FAD - Major OU Control Points and Strobes
G Cycles
GS
GE
GA
GO
GN
GF
¢ZDI0-7/M64-71
¢ZDI8-35/M0-28
¢M0-63/H0-63
¢M64-71/D
$M ($DRDY)
$ START
$
¢AQ0-71/ZAQ0-71
¢ZAQ0-71/N0-71
¢D/E
¢D̄/ES, ¢CARRY/ES7
¢ES̄/G
¢N-4
¢ DECREMENT G
¢S0-71/N0-71
¢N+4
¢ INCREMENT G
REV
¢N-1
LSN
¢S0-71/AQ0-71
¢Ḡ/ES, ¢CARRY/ES7
¢ES/E
$N
$H
$E
$D
$G
$AQ

Note:
E = 00001011
AQ = 1011010⟶0
Y_{0-7} = 00001111
Y_{8-35} = 00010⟶0

Fig 12.3

FIG. 12.3.8.1 FAD Instruction, Timing Diagram

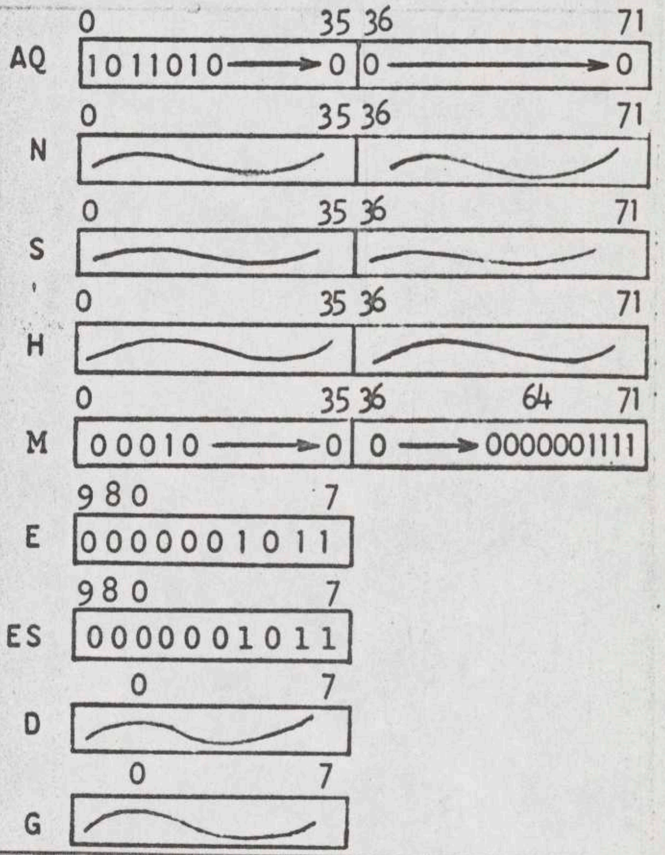
FAD
Floating Add Normalized

All registers are shown before the strobe at the end of the cycle.

GS

¢M_{0-63}/H_{0-63}
¢M_{64-71}/D
¢ Reset ZERØ
¢ Reset SIGN
$H, $N, $D

} Strobe on $


AQ
0
35 36
71
1011010→0
0→0
N
0
35 36
71
S
0
35 36
71
H
0
35 36
71
M
0
35 36
64
71
00010→0
0→0000001111
E
9 8 0
7
0000001011
ES
9 8 0
7
0000001011
D
0
7
G
0
7

GE

¢AQ_{0-71}/ZAQ_{0-71}
¢ZAQ_{0-71}/H_{0-71}
¢D/E
¢$\overline{D}$/ES
¢ CARRY/ES_7
¢ZAQ_{0-71}/N_{0-71} if ES_9
¢$\overline{ES}$/G if $0 > ES > -72$
$N, $E, $G } Strobe on $

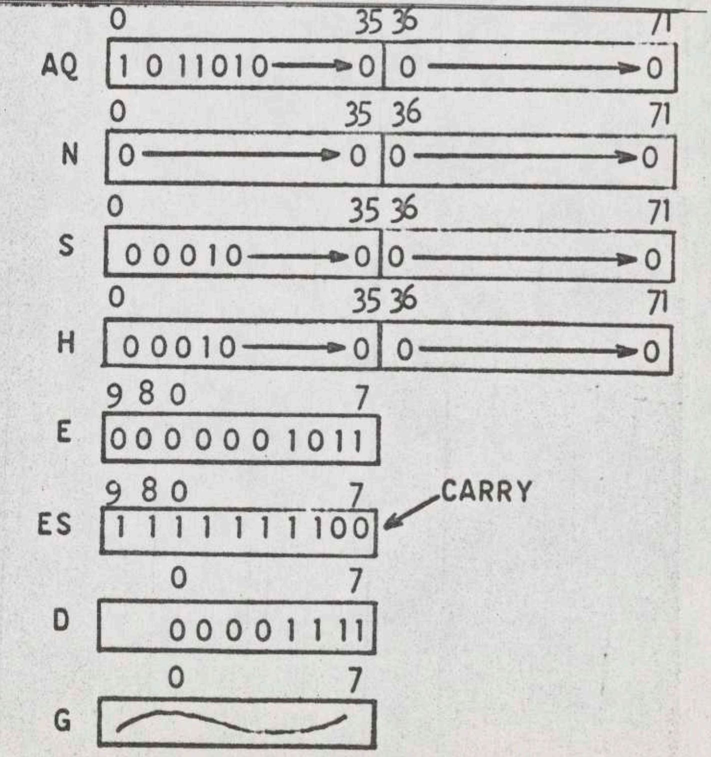
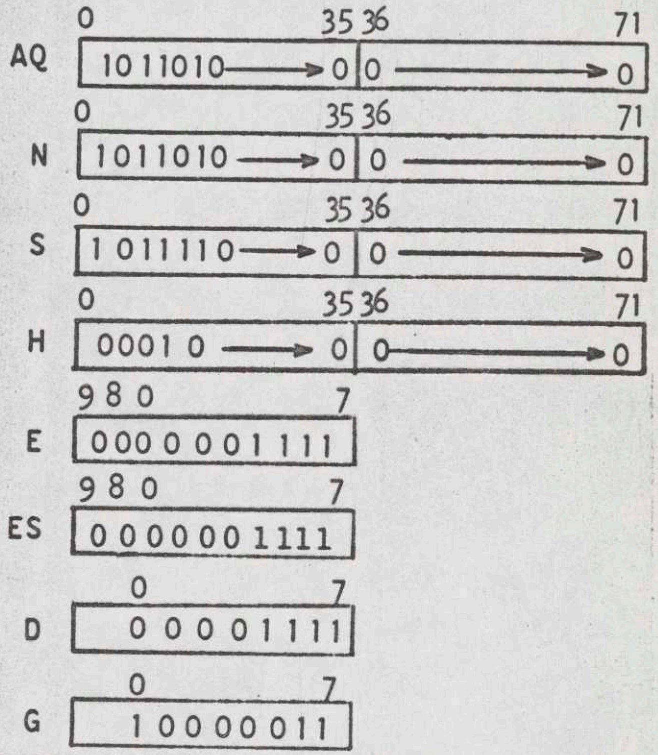

AQ
0
35 36
71
1011010→0
0→0
N
0
35 36
71
0→0
0→0
S
0
35 36
71
00010→0
0→0
H
0
35 36
71
00010→0
0→0
E
9 8 0
7
0000001011
ES
9 8 0
7
1111111100
CARRY
D
0
7
00001111
G
0
7

FIG. 12.3.8.1a Register Contents during GS, GF Cycles of FAD Instruction

GA

¢ N-4
LSA
¢ Decrement G by 4
\$N, \$G } Strobe on \$


0
35 36
71
AQ
1011010 → 0
0 → 0
0
35 36
71
N
1011010 → 0
0 → 0
0
35 36
71
S
1 011110 → 0
0 → 0
0
35 36
71
H
0001 0 → 0
0 → 0
9 8 0
7
E
000 0 001 111
9 8 0
7
ES
0 00000 1111
0
7
D
0 0 0 0 1 1 1 1
0
7
G
1 0 0 0 0 0 1 1

G0

¢ S_{0-71}/N_{0-71}
¢ SET SIGN if S_0
¢ Set ZER0 if S = 0
¢ Set CARRY if there is a carry out of S_0
\$H, \$N, \$G
} Strobe on \$

NOTE: CARRY will set

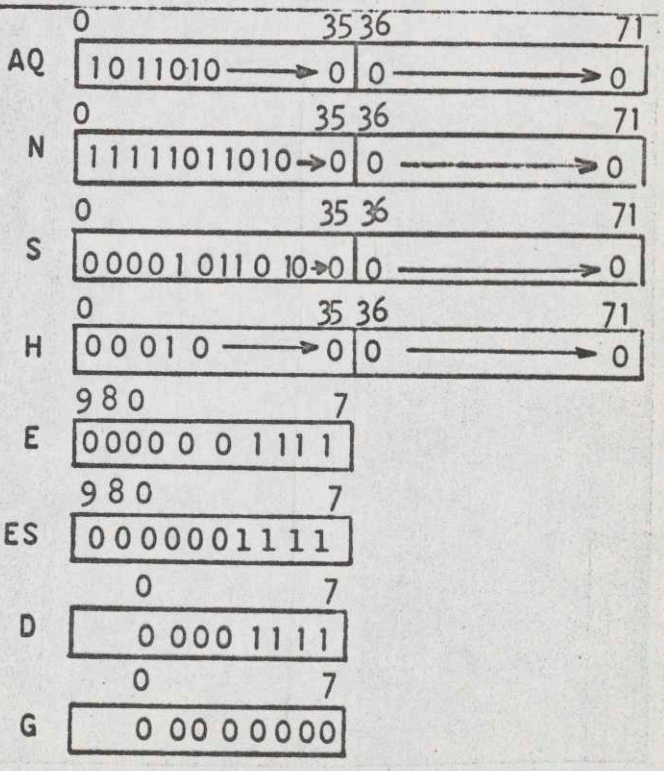
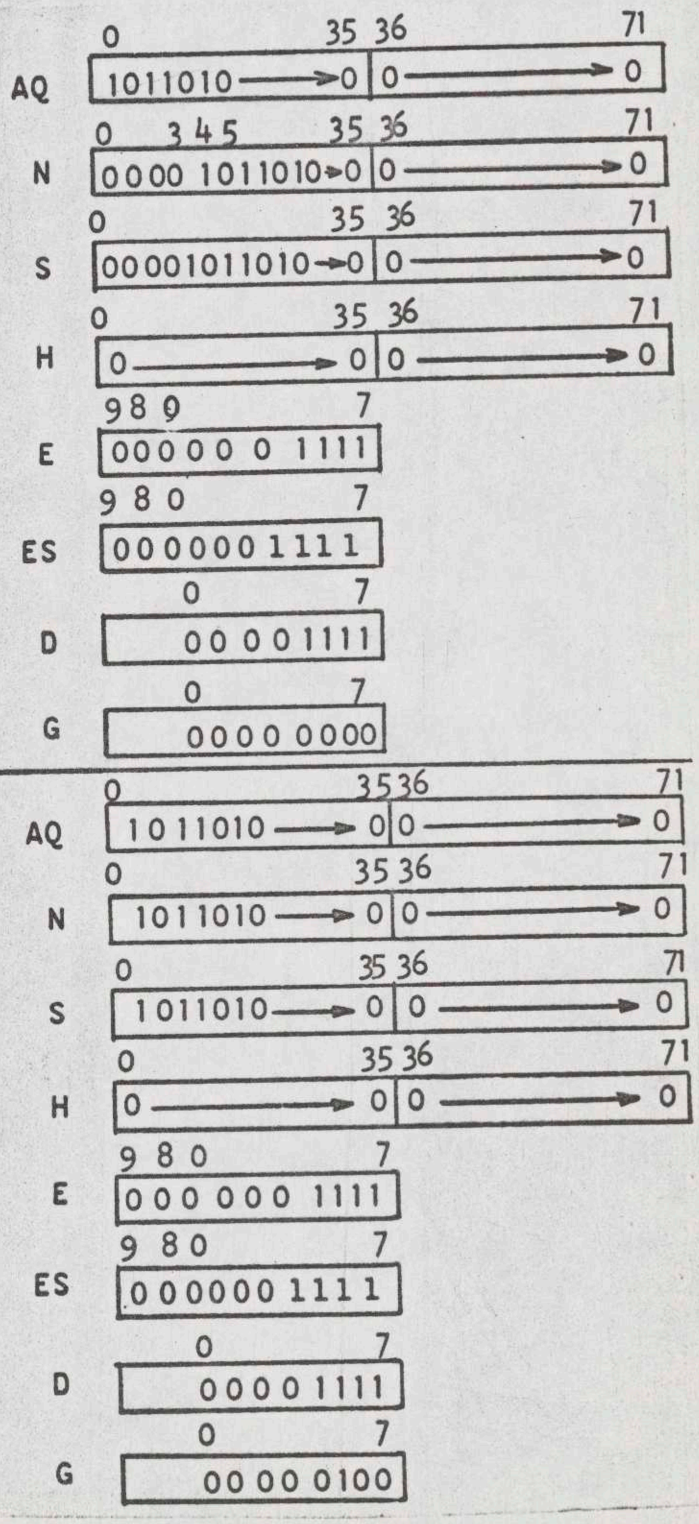

0
35 36
71
AQ
1011010 → 0
0 → 0
0
35 36
71
N
11111011010 → 0
0 → 0
0
35 36
71
S
00001 0110 10 → 0
0 → 0
0
35 36
71
H
0001 0 → 0
0 → 0
9 8 0
7
E
0000 0 0 111 1
9 8 0
7
ES
0000001111
0
7
D
0 000 1111
0
7
G
0 00 00000

Fig 12.3.8.1b

FIG. 12.3.8.1b Register Contents during GA, G0 Cycles of FAD Instruction


AQ
0 35 36 71
1011010 → 0 | 0 → 0
N
0 3 4 5 35 36 71
0000 1011010 → 0 | 0 → 0
S
0 35 36 71
00001011010 → 0 | 0 → 0
H
0 35 36 71
0 → 0 | 0 → 0
E
9 8 0 7
00000 0 1111
ES
9 8 0 7
0000001111
D
0 7
00 0 0 1111
G
0 7
0000 0000
AQ
0 35 36 71
1011010 → 0 | 0 → 0
N
0 35 36 71
1011010 → 0 | 0 → 0
S
0 35 36 71
1011010 → 0 | 0 → 0
H
0 35 36 71
0 → 0 | 0 → 0
E
9 8 0 7
000 000 1111
ES
9 8 0 7
0000001111
D
0 7
00001111
G
0 7
00 00 0100

__GN__ (first)

¢ N+4
¢ Increment G by 4
¢ Set REV } Strobe on $
$N, $G

NOTE: REV will set

__GN__ (second)

¢ N-1
¢ Decrement G by 1
LSN
¢ Reset REV } Strobe on $
$N, $G

NOTE: Last GN cycle

Fig 12.3.8.1c

FIG. 12.3.8.1c Register Contents during GN Cycle of FAD Instruction

GF

¢S_{0-71}/AQ_{0-71}

¢$\overline{G}$/ES

¢ CARRY to ES_7

¢ ES/E

$AQ, $E } Strobe an $

Contents of the registers upon completion of the FAD operation.

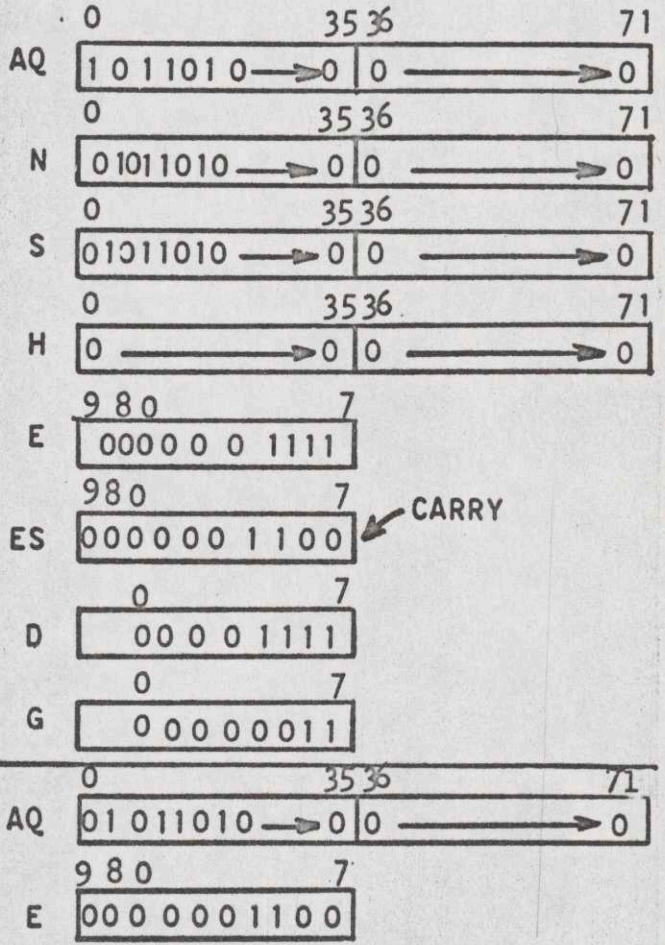

AQ
0 35 36 71
1011010 → 0 | 0 → 0
N
0 35 36 71
01011010 → 0 | 0 → 0
S
0 35 36 71
01011010 → 0 | 0 → 0
H
0 35 36 71
0 → 0 | 0 → 0
E
9 8 0 7
0000 0 0 1111
ES
9 8 0 7
000000 1100
CARRY
D
0 7
00 0 0 1111
G
0 7
0 0000011
AQ
0 35 36 71
01011010 → 0 | 0 → 0
E
9 8 0 7
000 0001100

Fig 12.3.8.1D

FIG. 12.3.8.1d Register Contents during GF Cycle of FAD Instruction

12.3.8.1 - contd

Unnormalized

Within this grouping are the eight instructions ADE, FCMG, FCMP, FLD, FSZN, LDE, UFA, and UFS.

Example 1: Load (FLD)

During this instruction the contents of a memory location Y are loaded into the Floating Register. That is, the exponent (Y_{0-7}) is loaded into the E-register and the mantissa (Y_{8-35}) is loaded into the A-register bits 0-27. The remaining bits of the A-register and all bits of the Q-register are cleared to zero. The presence of the FLD instruction code on the ZOR lines together with the Double Precision flip-flop (DPFF) reset and the Multiply flip-flop (DIC) reset, enables the IM-switch. The control points ¢ZDI_{0-7}/M_{64-71} and ¢ZDI_{8-35}/M_{0-27} are enabled presenting the operand lines (ZDI bus) to the proper M-register inputs. When the operand ready strobe (\$ DRDY) is received from the CU, DPFF reset allows the M-register strobe ($\$M_{0-71}$) to be generated and strobe the FLD operand into the M-register. The \$DRDY pulse is turned around as \$ YRY and returned to the CU to signal that a new instruction may be prepared. \$YRY also sets the operand ready flip-flop (YRY) in the OU. When YRY is set, the conditions exist to allow \$ START to be generated by a last of three pulse detectors. \$ START sets the OU Busy flip-flop (OUB), resets YRY and DDF, and transfers the operation code from the 0-register to the C-register. It also enters the timing pulse generator delay line network.

GS

When the $ START pulse propagates the medium delay elements, the operate strobe ($) occurs. This causes the following events:

1) A $ H which will strobe the FLD mantissa into the H-register bits 0-27 and clear all other bits to zero.

2) A $ D which will strobe the FLD exponent into the D-register.

3) A $ N which will strobe all zeros into the N-register since the IN-switch is not enabled.

4) The ZERO and SIGN indicators to reset.

5) ORB to reset, indicating that the O-register is free to receive the operation designator for the next instruction.

6) GS to reset and GF to set.

GF

The $ pulse will re-enter the timing pulse generator delay line network. GF being set will delay the pulse 400 nanoseconds before another $ is generated.

During this 400 nanoseconds the contents of H and N are added together in the S-adder. The add function takes place because none of the K flip-flops are set. The addition of H (mantissa) and N (zeros) produces the mantissa at the S-adder output. The following control points will also be enabled:

1) ¢ S_{0-35}/AQ_{0-35}: Transfer the output of the upper portion of the S-adder to the A-register.

2) ¢ S_{35-71}/AQ_{35-71}: Transfer the contents of the lower portion of the S-adder to the Q-register.

3) ¢ D/E: Transfer the contents of the D-register to the E-register.

The next $ strobe will cause the following events:

1) $ AQ which will strobe the mantissa into AQ_{0-27} and zeros into AQ_{28-71}.

2) $ E which will strobe the exponent into the E-register.

3) ZERO to set if the contents of AQ = zero.

4) SIGN to set if bit position zero of AQ is a "one"

5) GF to reset.

6) OUBF to reset if no other instruction has been initiated.

$ will generate a $ LAST, which is sent to the CU to indicate completion of the FLD instruction.

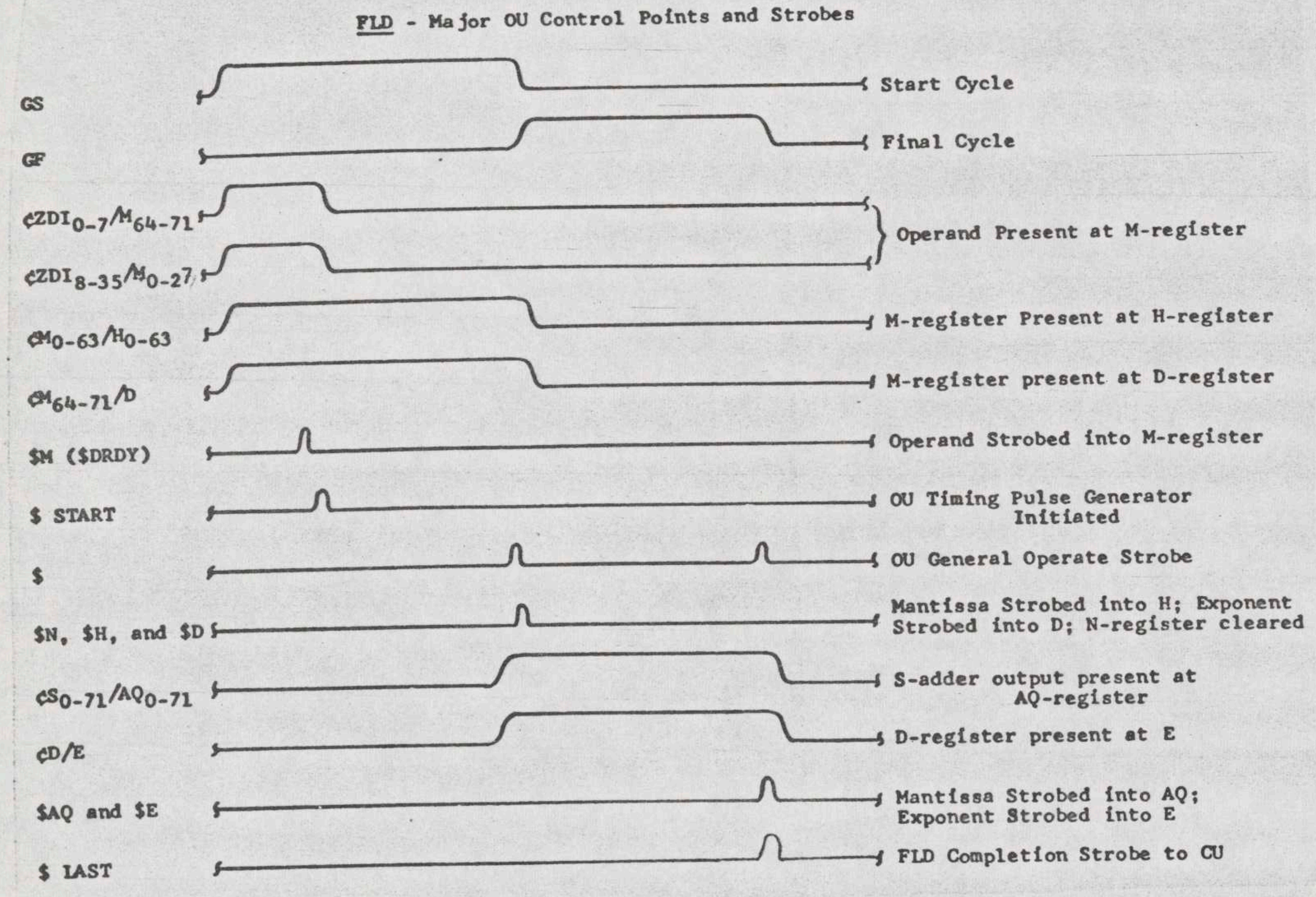

FLD - Major OU Control Points and Strobes
GS
GF
$¢ZDI_{0-7}/M_{64-71}$
$¢ZDI_{8-35}/M_{0-27}$
$¢M_{0-63}/H_{0-63}$
$¢M_{64-71}/D$
$M ($DRDY)
$ START
$
$N, $H, and $D
$¢S_{0-71}/AQ_{0-71}$
¢D/E
$AQ and $E
$ LAST
Start Cycle
Final Cycle
Operand Present at M-register
M-register Present at H-register
M-register present at D-register
Operand Strobed into M-register
OU Timing Pulse Generator Initiated
OU General Operate Strobe
Mantissa Strobed into H; Exponent Strobed into D; N-register cleared
S-adder output present at AQ-register
D-register present at E
Mantissa Strobed into AQ; Exponent Strobed into E
FLD Completion Strobe to CU

Fig 12.3.55

FIG. 12.3.8.1e FLD Instruction, Timing Diagram

Example 2: Add to Exponent Register (ADE)

During this instruction the contents of bits 0-7 of a memory location Y are added to the contents of the E-register and the results are placed in E. The presence of the ADE instruction code on the ZOR lines together with DPFF reset and DIC reset enables the IM-switch. The control points ¢ZDI_{0-7}/M_{64-71} and ZDI_{8-35}/M_{0-28} are enabled presenting the operand lines (ZDI bus) to the proper M-register inputs. When the operand ready strobe ($ DRDY) is received from the CU, DPFF reset allows the M-register strobe (\$ M_{0-71}) to be generated and strobe the ADE operand into the M-register. The $ DRDY pulse is turned around as $ YRY and returned to the CU to signal that a new instruction may be prepared. $ YRY also sets YRY in the OU. When YRY is set the conditions exist to allow $ START to be generated by a last of three pulse detectors. $ START sets OUB, resets YRY and DDF, and transfers the operation code from the O-register to the C-register. It also enters the timing pulse generator delay line network.

GS

When the $ START pulse propagates the medium delay elements, the operate strobe ($) occurs. This causes the following events:

1) $D which will strobe the ADE exponent into the D-register.

2) ZERO and SIGN to reset.

3) ORB to reset, indicating that the O-register is free to receive the operation designator for the next instruction.

4) GS to reset and GF to set.

GF

The $ pulse will re-enter the timing pulse generator delay line network. GF being set will delay the pulse 400 nanoseconds before another $ pulse is generated.

During this 400 nanoseconds control point ¢ D/ES is enabled, which will cause the contents of the E-register and the D-register to be added in the ES-adder. At the same time control point ¢ ES/E is enabled, which will place the sum output of the ES-adder on the inputs to the E-register. The $ pulse causes the following events:

1) A $ E is generated, which will strobe the new exponent into the E-register.
2) EOFL to set if an exponent overflow occurred.
3) EUFL to set if an exponent underflow occurred.
4) GF to reset.
5) OUBF to reset if no other instruction has been initiated.

$ will generate a $ LAST, which is sent to the CU to indicate completion of the ADE instruction.

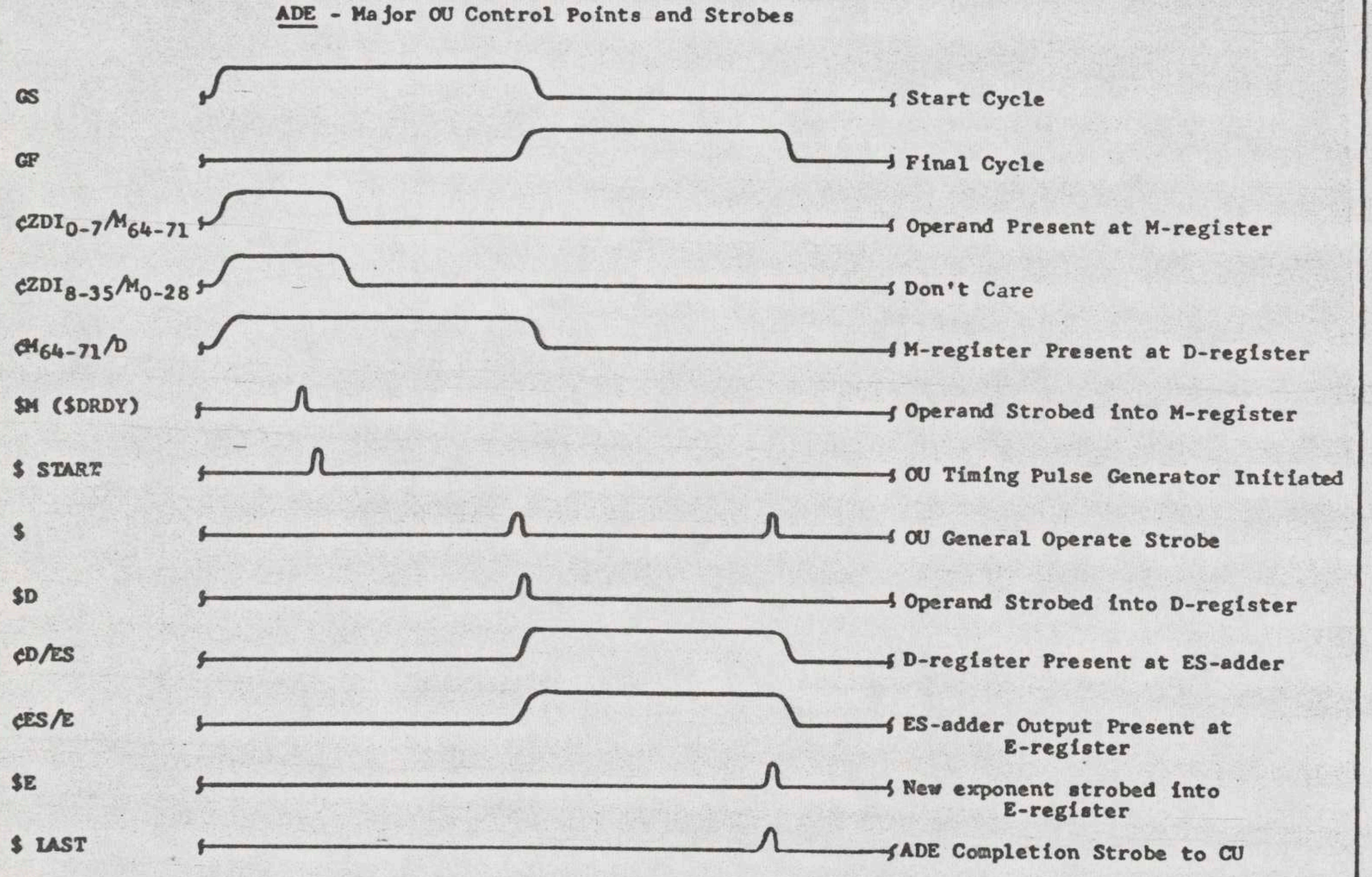

ADE - Major OU Control Points and Strobes
GS
GF
¢ZDI0-7/M64-71
¢ZDI8-35/M0-28
¢M64-71/D
$M ($DRDY)
$ START
$
$D
¢D/ES
¢ES/E
$E
$ LAST
Start Cycle
Final Cycle
Operand Present at M-register
Don't Care
M-register Present at D-register
Operand Strobed into M-register
OU Timing Pulse Generator Initiated
OU General Operate Strobe
Operand Strobed into D-register
D-register Present at ES-adder
ES-adder Output Present at E-register
New exponent strobed into E-register
ADE Completion Strobe to CU

Fig 12.3.8.1F

FIG. s12.3.8.1f ADE Instruction, Timing Diagram

Example 3: Compare (FCMG)

During this instruction the contents of a memory location Y are magnitude compared with the contents of the Floating Register (EAQ) and the ZERO and SIGN Indicators are set depending on the comparison. In general, this is accomplished by comparing the exponents of both numbers, select the number with the lesser exponent and shift its mantissa right as many places as the difference of the exponents states, then compare the absolute value of the mantissas, and set the indicators accordingly. The effective length of both mantiassas is 28 bits (including the sign). A compare is a subtract operation where the register contents are not altered. Since this operation is a compare magnitude, the operands to be compared must be of unlike signs. If the mantissas to be compared are of like signs, the memory mantissa must be two's complemented before the compare is actually made.

The data path from ZDI/M and the generation of $ START are the same as for the FLD instruction.

GS

During GS the signs of the mantissas (M_0 and A_0) are used to control the transfer of data from M to H. Because of the compare algorithm used, these signs must be opposite. Therefore, if these signs are not different, the IH-switch is enabled by ¢M_{0-71}/H_{0-71} and ¢M_{64-71}/H_{64-71}; if the signs are different, the IH-switch is enabled by $\$M_{0-63}/H_{0-63}$. At the same time ¢$M_{64-71}/D$ enabled causes the exponent to be presented at the inputs to the D-register.

When the $ START pulse propagates the medium delay elements, the operate strobe ($) occurs. This causes the following events:

1) $H is generated, which strobes the mantissa from M into H.

2) $N is generated, which clears N to zero.

3) $D is generated, which strobes the exponent from M into D.

4) If the signs of A and M are the same, K_{72} control flip-flop will be set to force an initial carry into the S-adder.

5) ZERO and SIGN to reset.

6) ORB to reset.

7) GS to reset and GE to set.

8) $ enters the timing pulse generator delay line network to produce another $ strobe in approximately 400 nanoseconds.

GE

During the GE cycle the two exponents are compared by subtracting D from E. The difference, if any, is stored in the G-counter. The results of this comparison will produce the following results:

1) If the difference is equal to zero, transfer the mantissa in AQ to N, leave H as is, and go to GF cycle.

2) If the difference is greater than +71, clear N, transfer the mantissa in AQ to H, and go to GF cycle.

3) If the difference is less than -71, clear N, leave H as is, and go to GF cycle.

4) If the difference is positive and less than +72, transfer the mantissa in AQ to H, store the difference in the G-counter and go to GA cycle.

5) If the difference is negative and greater than -72 transfer the mantissa in AQ to N, leave H as is, store the one's complement of the difference with the sign bit on in the G-counter and go to GA cycle.

The control lines that become enabled to accomplish these events are:

1) ¢AQ_{0-71}/ZAQ_{0-71} unconditionally

2) ¢ZAQ_{0-71}/H_{0-71} unconditionally

3) ¢$\overline{D_{0-7}}/ES_{0-9}$ unconditionally

4) ¢ CARRY/ES_7 unconditionally

5) *¢ZAQ_{0-35}/N_{0-35} if $[(ES_9) + (S = 0) + (ES = 0)] = 1$

*ZAQ bits 28 through 35 are forced to zero.

6) ¢S_{0-71}/N_{0-71} if $[\overline{(ES_9) + (S = 0) + (ES = 0)}] = 1$

7) ¢ES/G if $(ES_0)\ [(ES \neq 0)(/ES/<72)(ES_9 + S \neq 0)]$

8) ¢$\overline{ES}$/G if $(\overline{ES_0})\ [(ES \neq 0)(/ES/<72)(ES_9 + S \neq 0)]$

At the end of 400 nanoseconds a $ strobe will cause the following events:

1) A $ H to strobe the mantissa from AQ into H if $\overline{ES_9 + S = 0 + ES = 0}$

2) A \$ N to strobe a mantissa into N if $(/ES/ < 72) + (ES = 0) + (S = 0)(\overline{ES_9})$

3) A \$ G_{0-7} to strobe the proper count into the G-counter unconditionally.

4) LAFF to set if a maximum negative mantissa was two's complemented and transferred to N.

5) K_{72} to reset if the two's complemented mantissa was transferred to N.

6) GE to reset unconditionally.

7) GA to set if $(ES \neq 0)(/ES/ < 72)(ES_9 + S \neq 0)$.

8) GF to set if $(ES = 0) + (/ES/ > 71) + (\overline{ES}_9 \cdot S = 0)$.

9) \$ enters the timing pulse generator delay line network to produce another \$ strobe.

GA

During this cycle the mantissa in N is aligned so that its binary point falls at the same place as that of the mantissa in H. This is accomplished by shifting it right the number of places indicated by the G-counter. The control lines that become enabled to accomplish these events are;

1) ¢ N-16 (¢N_{0-55}/N_{16-71}, ¢N_0/N_{0-15}) if $(G > 14)$ ($\overline{\text{LAFF}}$)
2) ¢ N-4 (¢N_{0-67}/N_{4-71}, ¢N_0/N_{0-3}) if $(15 > G > 2)$($\overline{\text{LAFF}}$)
3) ¢ N-1 (¢N_{0-70}/ N_{1-71}, ¢N_0/N_0) if $(3 > G > 0)$($\overline{\text{LAFF}}$)
4) ¢ N-1 (¢N_{0-70}/N_{1-71}, ¢$\overline{N_0}/N_0$) if LAFF.
5) ¢ N+1 (¢N_{1-71}/N_{0-70}) if REV set.
6) LSA enabled if $(G = 1) + (G = 4) + (G = 16)$ + (REV)

At the end of approximately 230 nanoseconds a $ strobe will cause the following events:

1) A $ N to strobe the shifted contents of N back into N.
2) A $ G to decrement G by 16, if ¢N = 16, by 4 if ¢N-4, by 1 if ¢N-1, or decrement G to zero (0) if G = 1, 4, or 15.
3) LAFF to reset.
4) REV to set if $(G = 15) + (G = 3)$
5) GA to reset and GF to set if LSA enabled.
6) $ enters the timing pulse generator delay line network to produce another $ strobe.

GA will be repeated as many times as required to align the mantissa as indicated by LSA enabled.

GF

During GF the contents of the N-and H-registers are subtracted in the S-adder.

On the occurance of the next \$ strobe, an indicator may be set depending upon the compare. If the mantissa in AQ is equal to the mantissa in Y, the ZERO Indicator is set. If the mantissa in AQ is greater than the mantissa in Y, no indicators are set. If the mantissa in AQ is smaller than the mantissa in Y, the SIGN Indicator is set. This \$ strobe also:

1) Resets the GF cycle flip-flop.

2) Resets the OUB flip-flop if no other instruction has been initiated.

3) Resets the K_{72} flip-flop.

4) Will generate a \$ LAST, which is sent to the CU to indicate completion of the FCMG instruction.

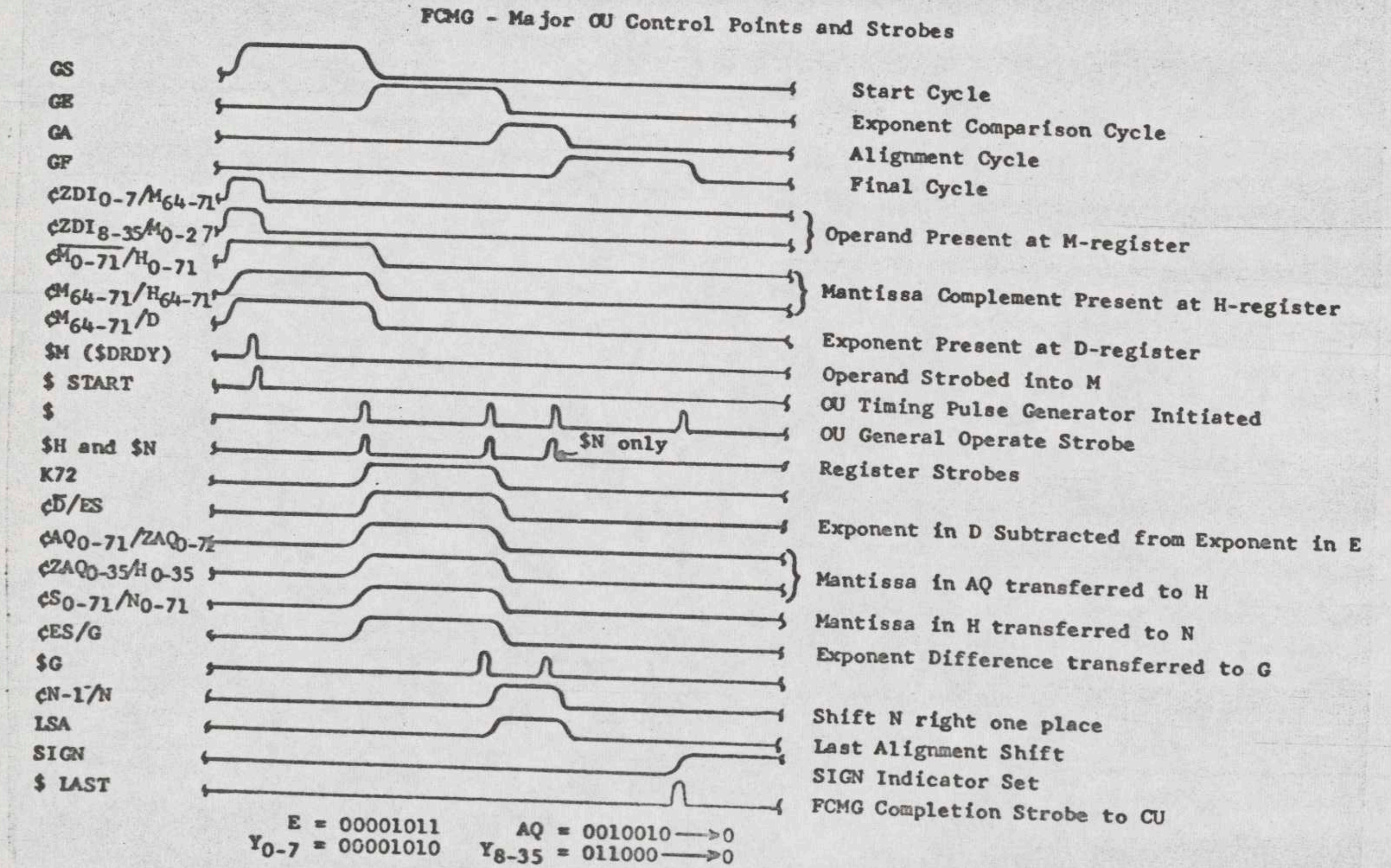

FCMG - Major OU Control Points and Strobes
GS
GE
GA
GF
¢ZDI0-7/M64-71
¢ZDI8-35/M0-27
¢M0-71/H0-71
¢M64-71/H64-71
¢M64-71/D
$M ($DRDY)
$ START
$
$H and $N
K72
¢D/ES
¢AQ0-71/ZAQ0-71
¢ZAQ0-35/H0-35
¢S0-71/N0-71
¢ES/G
$G
¢N-1/N
LSA
SIGN
$ LAST
$N only
Start Cycle
Exponent Comparison Cycle
Alignment Cycle
Final Cycle
Operand Present at M-register
Mantissa Complement Present at H-register
Exponent Present at D-register
Operand Strobed into M
OU Timing Pulse Generator Initiated
OU General Operate Strobe
Register Strobes
Exponent in D Subtracted from Exponent in E
Mantissa in AQ transferred to H
Mantissa in H transferred to N
Exponent Difference transferred to G
Shift N right one place
Last Alignment Shift
SIGN Indicator Set
FCMG Completion Strobe to CU
E = 00001011
Y0-7 = 00001010
AQ = 0010010 ⟶ 0
Y8-35 = 011000 ⟶ 0

FIG. 12.3.8.1g FCMG Instruction, Timing Diagram

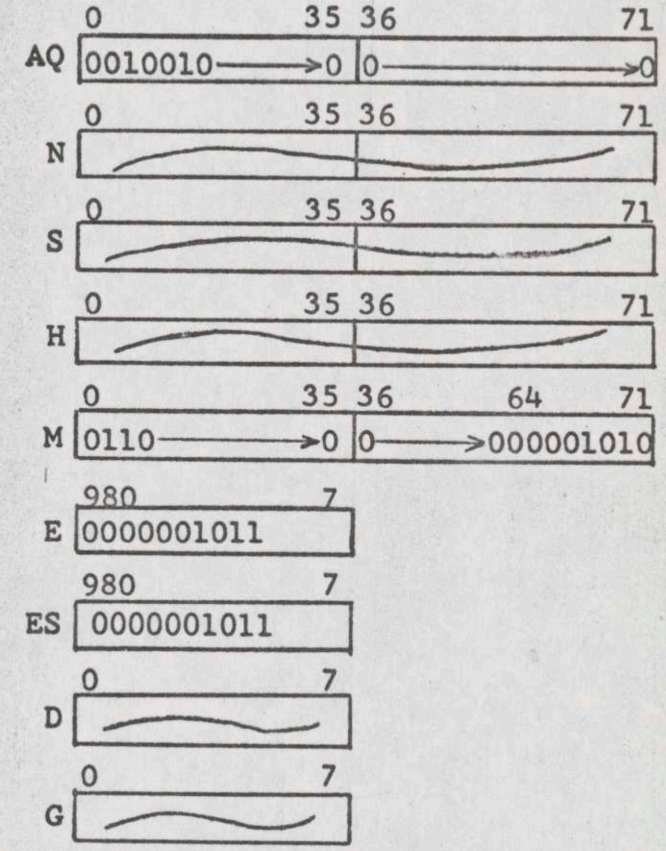
FCMG
Floating Compare Magnitude

All registers are shown before the strobe at the end of the cycle.

GS

¢$\overline{M_{0-71}}/H_{0-71}$
¢M_{64-71}/H_{64-71}
¢M_{64-71}/D

¢ Set K_{72}
¢ Reset ZERO
¢ Reset SIGN
\$H, \$N, \$D
} Strobe on \$


AQ
0 35 36 71
0010010 → 0
0 → 0
N
S
H
M
0 35 36 64 71
0110 → 0
0 → 000001010
E
980 7
0000001011
ES
0000001011
D
0 7
G

GE

¢AQ_{0-71}/ZAQ_{0-71}
¢ZAQ_{0-35}/H_{0-35}
¢$\overline{D}/ES$
¢$CARRY/ES_7$
¢S_{0-71}/N_{0-71}
¢ES/G
¢ Reset K_{72}
\$H, \$N, \$G
} Strobe on \$

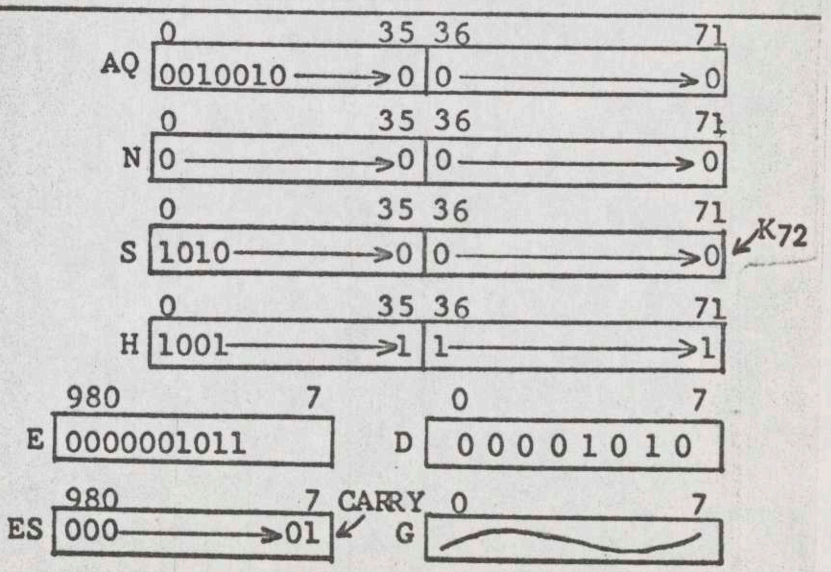
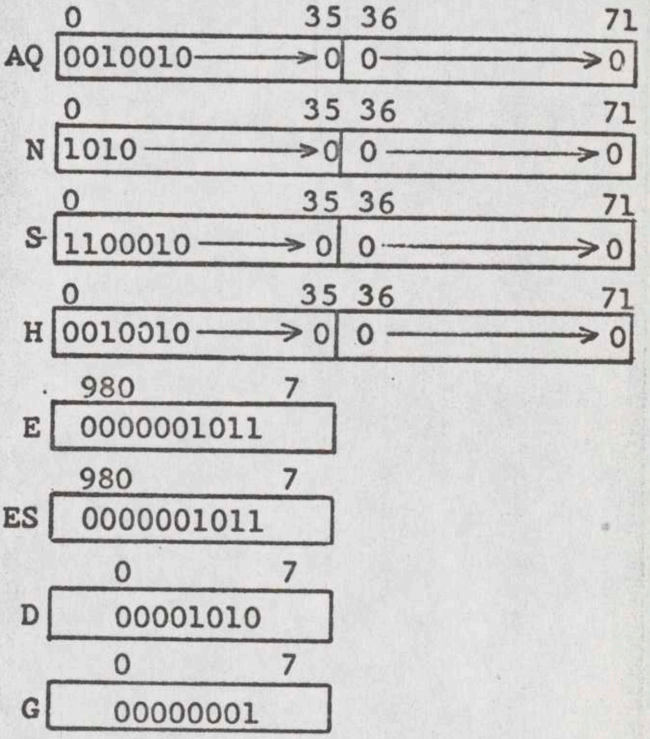

AQ
0 35 36 71
0010010 → 0
0 → 0
N
0 → 0
0 → 0
S
1010 → 0
0 → 0
K72
H
1001 → 1
1 → 1
E
980 7
0000001011
D
0 7
0 0 0 0 1 0 1 0
ES
000 → 01
CARRY
G

FIG. 12.3.8.1h Register Contents during GS, GE Cycles of FCMG Instruction

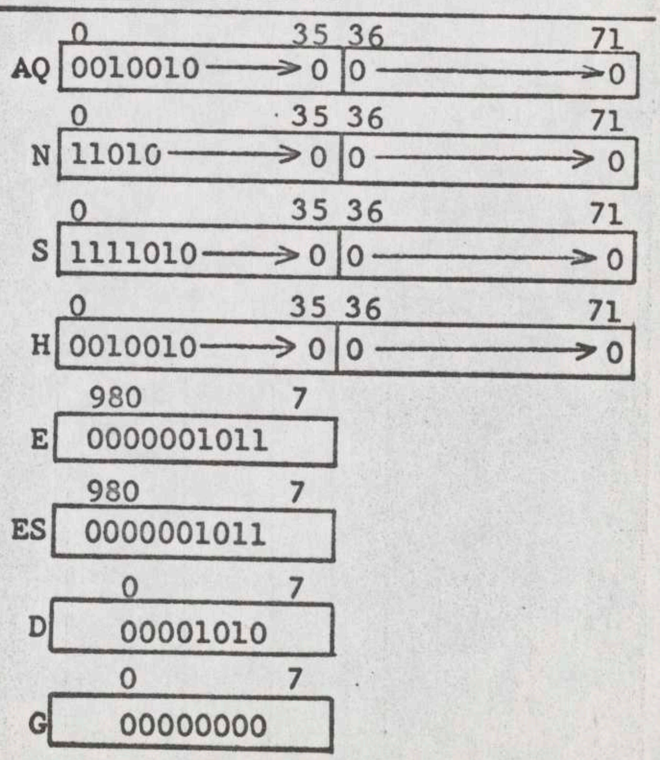
GA

¢N-1
¢ Decrement G by 1 } Strobe on $
$N $G


AQ 0 35 36 71
0010010 → 0 0 → 0
N 0 35 36 71
1010 → 0 0 → 0
S 0 35 36 71
1100010 → 0 0 → 0
H 0 35 36 71
0010010 → 0 0 → 0
E 980 7
0000001011
ES 980 7
0000001011
D 0 7
00001010
G 0 7
00000001

GF

¢ Set ZERO if S=0 } Strobe on $
¢ Set SIGN if S_0=1


AQ 0 35 36 71
0010010 → 0 0 → 0
N 0 35 36 71
11010 → 0 0 → 0
S 0 35 36 71
1111010 → 0 0 → 0
H 0 35 36 71
0010010 → 0 0 → 0
E 980 7
0000001011
ES 980 7
0000001011
D 0 7
00001010
G 0 7
00000000

Note: SIGN will set

Fig. 12.3.8.1i Register Contents during GA, GE Cycles of FCMG Instruction

Example 4: Unnormalized Floating Subtract (UFS)

During this instruction a floating point number in a memory location Y is subtracted from the contents of the Floating Register (EAQ). This is done by first comparing the exponents, then aligning the mantissas, then subtract the mantissas, and then store the difference in EAQ. In this instruction the resultant mantissa is not normalized.

The data path from ZDI/M and the generation of $ START are the same as for the FLD instruction.

GS

During GS the complement of the mantissa in M is transferred to H, the exponent in M is transferred to D, and the proper indicator and control flip-flops are set up. The control lines that become enabled to accomplish this are:

1) $\overline{\text{¢}M_{0-71}}/H_{0-71}$ and $\text{¢}M_{64-71}/H_{64-71}$

2) $\text{¢}M_{64-71}/D$

When the $ START propagates the medium delay elements the operate strobe ($) occurs. This causes the following events:

1) A $ H strobes the complement of the mantissa in M into H.

2) A $ N clears N to zero.

3) A $ D strobes the exponent in M into D.

4) ZERO and SIGN to reset.

5) K_{72} to set.

6) GS to reset and GE to set.

7) \$ enters the timing pulse generator delay line network to produce another \$ pulse in approximately 400 nanoseconds.

GE

During the GE cycle the two exponents are compared by subtracting D from E, the difference if any is stored in G, the mantissa is stored in H, and the larger exponent is stored in E. The control lines that become enabled to accomplish this are:

1) ¢AQ_{0-71}/ZAQ_{0-71} unconditionally

2) ¢ZAQ_{0-71}/H_{0-71} unconditionally

3) ¢D/E unconditionally

4) ¢$\overline{D}$/ES unconditionally

5) ¢ CARRY/ES_7 unconditionally

6) ¢ZAQ_{0-71}/N_{0-71} if $ES_9 + S = 0 + ES = 0$

7) ¢S_{0-71}/N_{0-71} if $\overline{ES_9 + S = 0 + ES = 0}$

8) ¢ES/G if $(ES_0)\,[(ES \neq 0)(/ES/<72)(ES_9 + S \neq 0)]$

9) ¢$\overline{ES}$/G if $(\overline{ES_0})\,[(ES \neq 0)(/ES/< 72)(ES_9 + S \neq 0)]$

At the end of 400 nanoseconds a \$ pulse will cause the following events:

1) A \$ H to strobe the mantissa from AQ into H if $\overline{ES_9 + S = 0 + ES = 0}$

2) A \$ N to strobe a mantissa into N if $/ES/< 72 + ES = 0 + (S = 0)(\overline{ES}_9)$

3) A \$ E to strobe the exponent from D to E if ES_9

4) A \$ G to strobe the proper count into G_{0-7} unconditionally

5) LAFF to set if a maximum negative number was two's complemented and transferred to N

6) K_{72} to reset if the two's complemented mantissa was transferred to N

7) GE to reset

8) GO to set if $(ES = 0) + (/ES/ > 71) + (ES_9)(S = 0)$

9) GA to set if $(ES \neq 0)(/ES/ < 72)(ES_9 + S \neq 0)$

10) \$ enters the timing pulse generator delay line network to produce another \$ pulse.

GA

During this cycle the mantissa in N is aligned so that its binary point falls at the same place as that of the mantissa in H. This is accomplished by shifting N right the number of places indicated by the G-counter. The control lines that become enabled to accomplish this are:

1) ¢N-16 (¢N_{0-55}/N_{16-71}, ¢N_0/N_{0-15}) if $(G>14)(\overline{LAFF})$

2) ¢N-4 (¢N_{0-67}/N_{4-71}, ¢N_0/N_{0-3}) if $(15>G>2)(\overline{LAFF})$

3) ¢N-1 (¢N_{0-70}/N_{1-71}, ¢$N_0/0$) if $(3>G>0)(\overline{LAFF})$

4) ¢N-1 (¢N_{0-70}/N_{1-71}, $\overline{\text{¢}N_0}/N_0$) if LAFF

5) ¢N+1 (¢N_{1-71}/N_{0-70}) if REV

6) LSA enabled if (G = 1) + (G = 4) + (G = 16) + (REV)

At the end of approximately 230 nanoseconds a $ pulse will cause the following events:

1) A $ N to strobe the shifted contents of N back into N.

2) A $ G to decrement G by 16 if ¢N-16, by 4 if ¢N-4, by 1 if ¢N-1, or increment G by 1 if ¢N + 1.

3) REV to set if (G = 15) + (G = 3).

4) GA to reset and GO to set if LSA enabled.

5) $ enters the timing pulse generator delay line network to produce another $ pulse.

GA will be repeated as many times as required to align the mantissa as indicated by LSA enabled.

GO

During this cycle the complemented mantissa is added to the other mantissa in the S-adder producing the difference. This difference is stored in N, H and G are cleared to zero, and the proper indicators are set. The control line that becomes enabled to accomplish this is:

¢S_{0-71}/N_{0-71} to transfer the output of the S-adder to N.

At the end of 400 nanoseconds a $ pulse will cause the following events:

1) A $ H that clears H to zero.
2) A $ N that strobes the mantissa difference into N.
3) REV to set if an overflow occurred.
4) SIGN to set if the difference is zero.
6) CARRY to set if there is a carry out of bit position zero.
7) CARRY to reset if there is no carry out of bit position zero.
8) K_{72} to reset.
9) GO to reset.
10) GN to set if an overflow occurred.
11) GF to set if no overflow occurred.
12) $ enters the timing pulse generator delay line network to produce another $ pulse.

GN

During this cycle a mantissa that has overflowed is shifted one place right and the G-counter is decremented by one. The control line that becomes enabled to accomplish this is:

¢ N-1 $(¢N_{0-70}/N_1/N_{71}, ¢\overline{N_0}/N_0)$

At the end of 230 nanoseconds a $ pulse will cause the following events:

1) A $ N to strobe the shifted contents of N back into N.
2) A $ G to decrement the G-counter by one.
3) REV to reset.
4) GN to reset and GF to set.
5) $ enters the timing pulse generator delay line network to produce another $ pulse.

GF

During this cycle the mantissa in N is transferred to AQ, the count in G is subtracted from the exponent in E, and EOFL is set if required. If the mantissa is zero, the exponent will be forced to -128. The control lines that become enabled to accomplish this are:

1) $¢S_{0-71}/AQ_{0-71}$
2) ¢G/ES and ¢ CARRY/ES_7

3) ¢ES/E if mantissa $\neq$ zero

4) ¢-2^7 /E if mantissa = zero

At the end of 400 nanoseconds a $ pulse will cause the following events to happen:

1) A $ AQ to strobe the mantissa into AQ.

2) A $ E to strobe the exponent into E.

3) EOFL to set if the subtraction of G from E caused the exponent to exceed +127.

4) GF to reset.

5) OUBF to reset if no other instruction has been initiated.

6) $ will generate a $ LAST, which is sent to the CU to indicate completion of the UFS instruction.

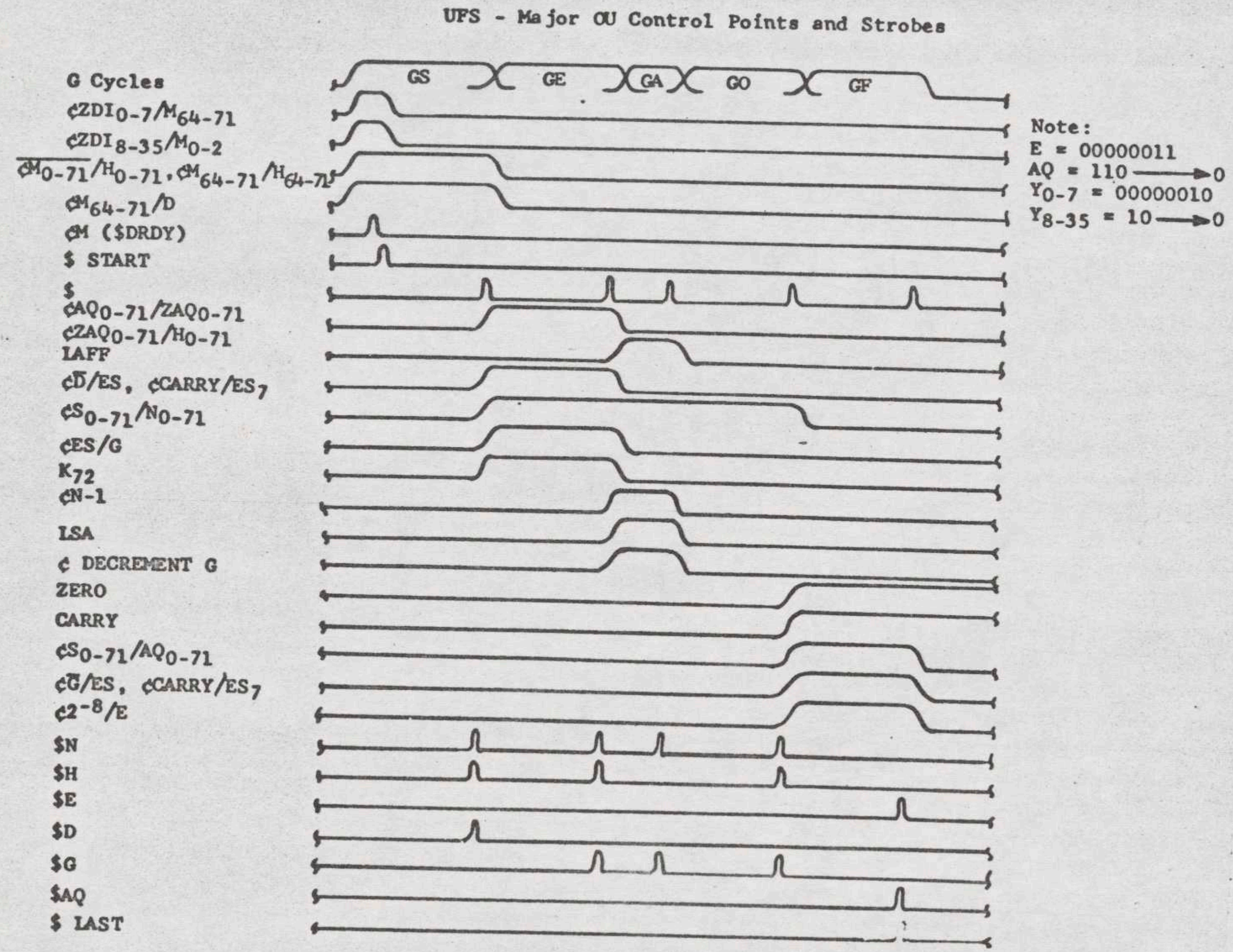

UFS - Major OU Control Points and Strobes
G Cycles
GS
GE
GA
GO
GF
¢ZDI0-7/M64-71
¢ZDI8-35/M0-2
¢M0-71/H0-71, ¢M64-71/H64-71
¢M64-71/D
¢M ($DRDY)
$ START
$
¢AQ0-71/ZAQ0-71
¢ZAQ0-71/H0-71
LAFF
¢D̄/ES, ¢CARRY/ES7
¢S0-71/N0-71
¢ES/G
K72
¢N-1
LSA
¢ DECREMENT G
ZERO
CARRY
¢S0-71/AQ0-71
¢Ḡ/ES, ¢CARRY/ES7
¢2^-8/E
$N
$H
$E
$D
$G
$AQ
$ LAST
Note:
E = 00000011
AQ = 110 → 0
Y0-7 = 00000010
Y8-35 = 10 → 0

Fig 12.3.8.1j

FIG. 12.3.8.1j UFS Instruction, Timing Diagram

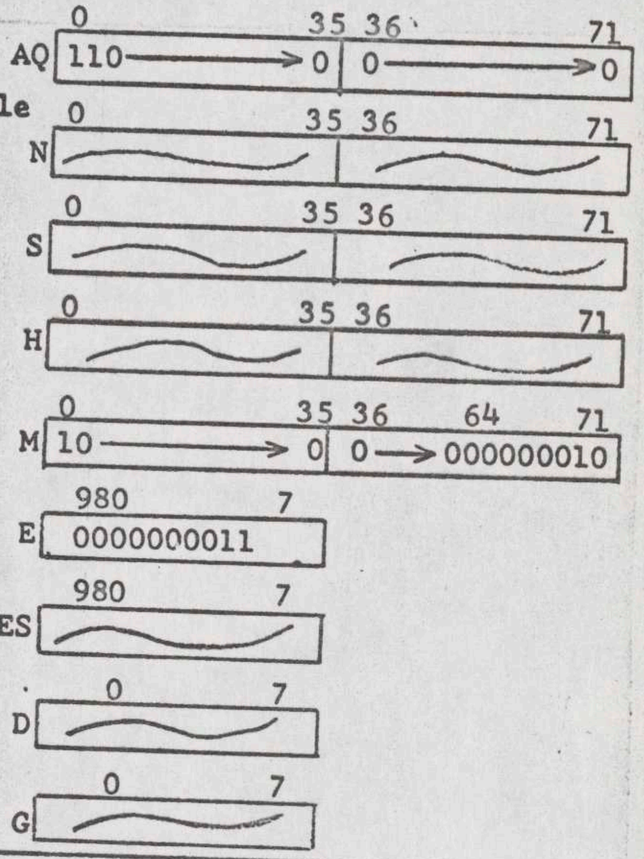
UFS
Unnormalized Floating Subtract

All registers are shown before the strobe at the end of the cycle

GS

$¢M_{0-71}/H_{0-71}$
$¢M_{64-71}/H_{64-71}$
$¢M_{64-71}/D$

$$\left.\begin{array}{l}\text{¢ Reset ZERO}\\ \text{¢ Reset SIGN}\\ \text{¢ Set } K_{72}\\ \text{\$H, \$N, \$D}\end{array}\right\}\begin{array}{l}\text{Strobe}\\ \text{on \$}\end{array}$$


0
35 36
71
AQ
110 → 0
0 → 0
N
S
H
M
10 → 0
0 → 000000010
64
E
980
7
0000000011
ES
D
G

GE

$¢AQ_{0-71}/ZAQ_{0-71}$
$¢ZAQ_{0-71}/H_{0-71}$
$¢\underline{D}/E$
$¢\overline{D}/ES$
$¢\ CARRY/ES_7$
$¢S_{0-71}/N_{0-71}$ of $\overline{ES_9}$
¢ES/G if 0<ES<72

$$\left.\begin{array}{l}\text{¢ Set LAFF}\\ \text{¢ Reset } K_{72}\\ \text{\$H, \$N, \$G}\end{array}\right\}\begin{array}{l}\text{Strobe}\\ \text{on \$}\end{array}$$

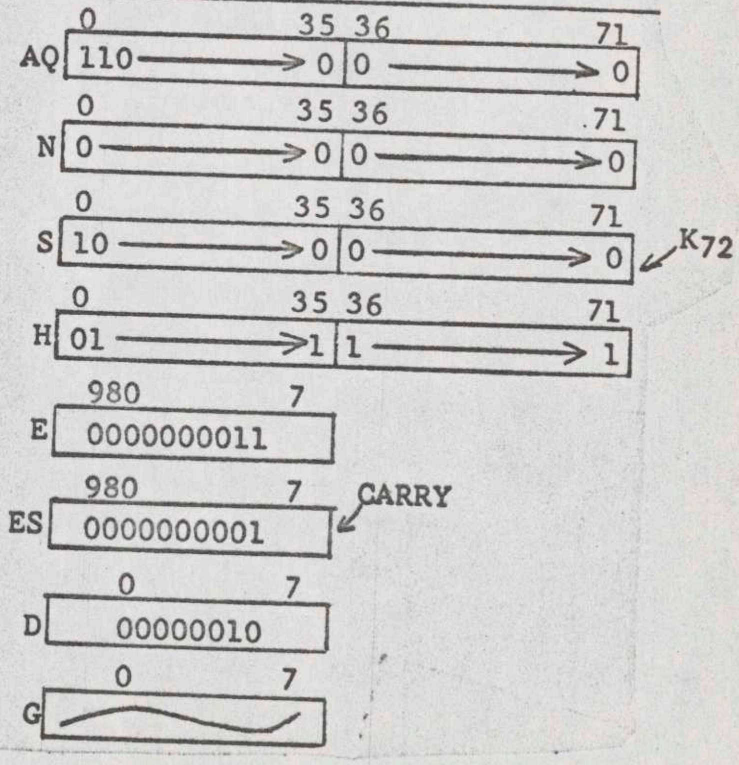
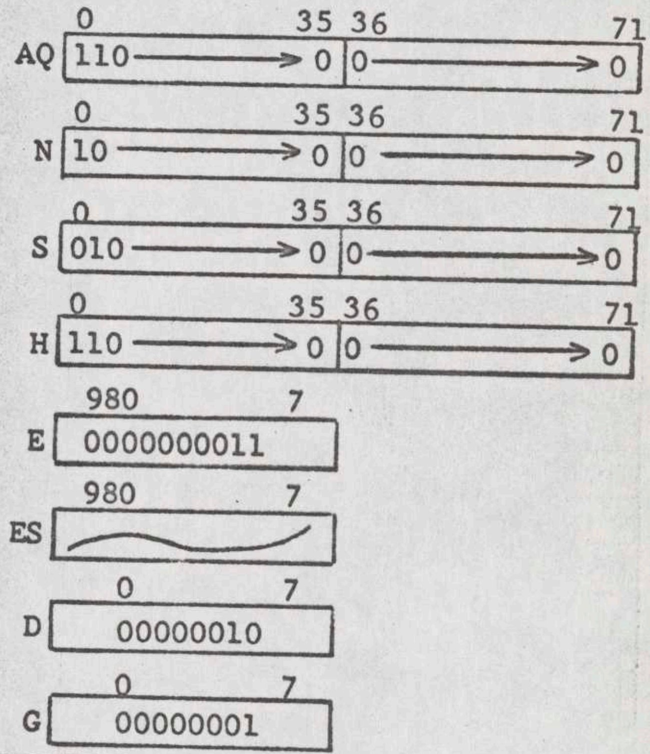

0
35 36
71
AQ
110 → 0
0 → 0
N
0 → 0
0 → 0
S
10 → 0
0 → 0
K72
H
01 → 1
1 → 1
E
980
7
0000000011
ES
0000000001
CARRY
D
00000010
G

FIG. 12.3.8.1k Register Contents during GS, GE Cycles of UFS Instruction

GA

¢N-1 ($¢N_{0-70}/N_{1-71}$, $¢\overline{N_0}/N_0$)
¢ Reset LAFF
LSA
¢ Decrement G by 1
\$N, \$G } Strobe on \$

Note:
Only one GA cycle


AQ 0 110 → 35 0 | 36 0 → 71 0
N 0 10 → 35 0 | 36 0 → 71 0
S 0 010 → 35 0 | 36 0 → 71 0
H 0 110 → 35 0 | 36 0 → 71 0
E 980 7 0000000011
ES 980 7
D 0 7 00000010
G 0 7 00000001

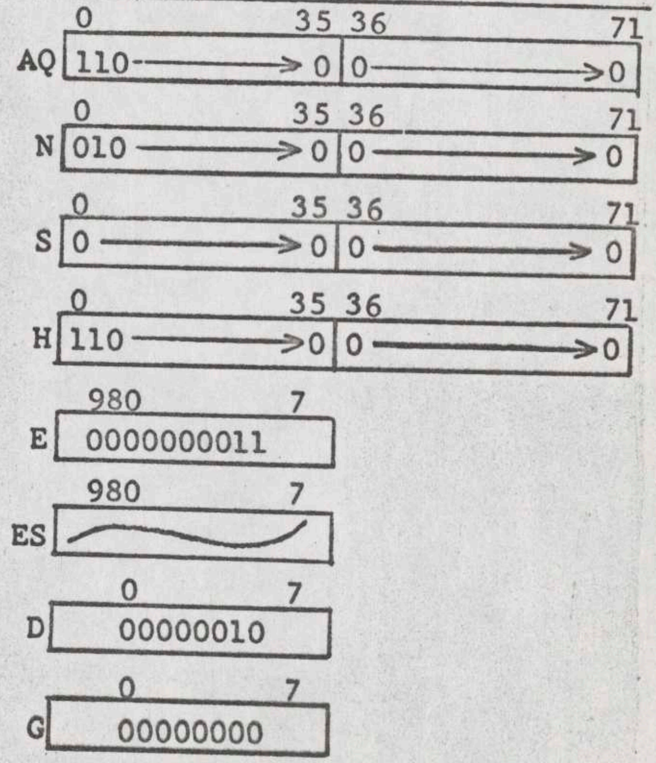
GO

$¢S_{0-71}/N_{0-71}$
¢ Set ZERO if S=0
¢ Set CARRY if there is a carry out of S_0
\$H, \$N, \$G
} Strobe on \$

Note:
ZERO and CARRY will set

FIG. 12.3.8.1m Register Contents during GA, GO Cycles of UFS Instruction

Fig 12.3.8.1m


AQ 0 110 → 35 0 | 36 0 → 71 0
N 0 010 → 35 0 | 36 0 → 71 0
S 0 0 → 35 0 | 36 0 → 71 0
H 0 110 → 35 0 | 36 0 → 71 0
E 980 7 0000000011
ES 980 7
D 0 7 00000010
G 0 7 00000000

GF

¢S_{0-71}/AQ_{0-71}
¢$\overline{G}$/ES
¢ CARRY/ES_7
¢-2^7/E if S=0
$AQ, $E } strobe on $

Contents of the registers upon completion of the UFS operation

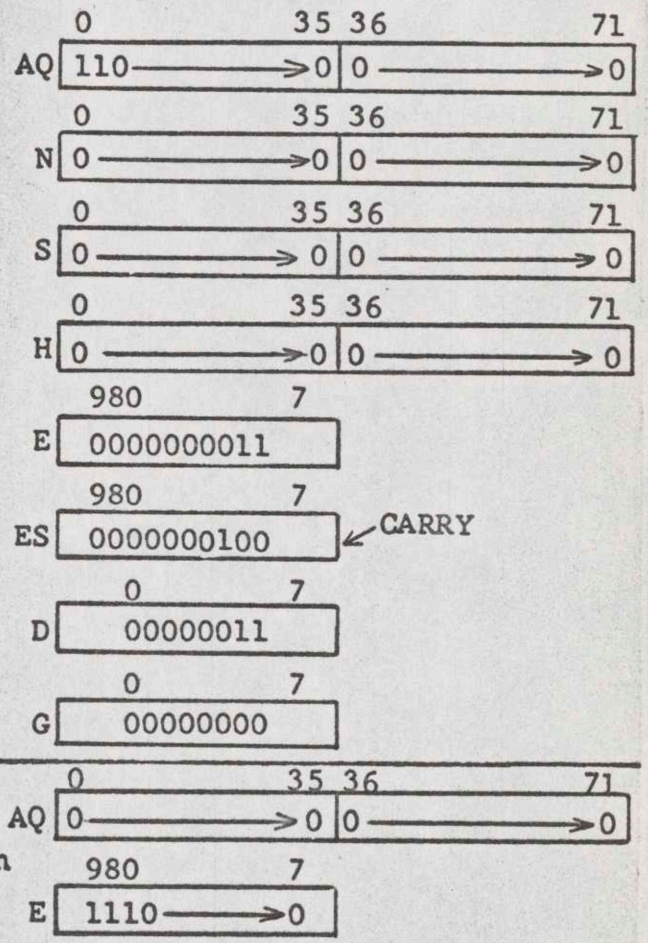

0
35 36
71
AQ
110 → 0
0 → 0
N
0 → 0
0 → 0
S
0 → 0
0 → 0
H
0 → 0
0 → 0
980
7
E
0000000011
ES
0000000100
CARRY
0
7
D
00000011
G
00000000
AQ
0 → 0
0 → 0
E
1110 → 0

FIG. 12.3.8.1n Register Contents during GF Cycle of UFS Instruction

12.3.8.2 Double Precision

The instructions comprising the double precision floating point add group can be defined by two classifications:

1) Instructions requiring normalization (DFAD and DFSB).

2) Instructions that do not require normalization (DFCMG, DFCMP, DFLD, DUFA, and DUFS).

The only difference in the OU between these instructions and the corresponding single precision instructions in the floating point add group is that these instructions manipulate a mantissa of 64 bits instead of 28 bits, and two \$DRDYs are required instead of one.

Normalized

Under this heading fall the two instructions DFAD and DFSB.

Example 1: Subtract (DFSB)

During this instruction a double precision floating point number in a memory location Y and Y + 1 is subtracted from the contents of the Floating Register (EAQ). This is done by first comparing the exponents, then aligning the mantissas, then subtracting the mantissas, then normalize the difference, and then store the difference in EAQ.

The IM-switch is enabled by the presence of the DFSB decode together with DPFF, BCFF, and DIC all reset. The control points ¢ZDI_{0-7}/M_{64-71} and ¢ZDI_{8-35}/M_{0-27} are enabled to allow the ZDI bus to be present at the proper M-register inputs. When the first \$DRDY is received from the CU the Double Precision flip-flop (DPFF) not yet being set allows

$\$M_{0-71}$ to be generated and strobe the exponent and most significant 28 bits of the mantissa into the M-register. This \$DRDY will set DPFF, but will not generate a \$ START. On the setting of DPFF, the IM-switch becomes enabled by $¢ZDI_{0-7}/M_{28-35}$ and $¢ZDI_{8-35}/M_{35-63}$.

About 250 nanoseconds after the first \$DRDY another \$DRDY will be received. When the second \$DRDY is received DPFF set allows $\$M_{28-63}$ to be generated and strobe the least significant 36 bits of the mantissa into the M-register. This \$DRDY pulse resets DPFF and is turned around as \$YRY and returned to the CU to signal that a new instruction may be prepared. \$YRY sets the operand ready flip-flop (YRY) in the OU. When YRY is set, the conditions exist to allow a \$START which sets OUB, resets YRY and DDF, and transfers the operation code from the O-register to the C-register. It also enters the timing pulse generator delay line network.

GS

During this cycle the complement of the mantissa im M is transferred to H, the exponent in M is transferred to D, and the proper indicator and control flip-flops are set up. The control lines that become enabled to accomplish this are:

1) $¢\overline{M_{0-71}}/H_{0-71}$ and $¢M_{64-71}/H_{64-71}$.

2) $¢M_{64-71}/D$

When the \$START propagates the medium delay elements the operate strobe (\$) occurs. This pulse causes the following events:

1) A \$H to strobe the complement of the mantissa in M into H.

2) A \$N to clear N to zero.

3) A $D to strobe the exponent in M into D.

4) ZERO and SIGN to reset.

5) K_{72} to set.

6) GS to reset and GE to set.

7) $ enters the timing pulse generator delay line network to produce another $ pulse.

GE

During this cycle the two exponents are compared by subtracting D from E, the difference if any is stored in G. the mantissa with the lesser exponent is stored in N, the other mantissa is stored in H, and the larger exponent is stored in E. The control lines that become enabled to accomplish this are:

1) $¢AQ_{0-71}/ZAQ_{0-71}$ unconditionally

2) $¢ZAQ_{0-71}/H_{0-71}$ unconditionally

3) ¢D/E unconditionally. unconditionally

4) $¢\overline{D}/ES$ and $¢CARRY/ES_7$ unconditionally

5) $¢ZAQ_{0-71}/N_{0-71}$ if $ES_9 + S = 0 + ES = 0.$

6) $¢S_{0-71}/N_{0-71}$ if $\overline{ES_9 + S = 0 + ES = 0.}$

7) ¢ES/G if (ES_0) $(ES \neq 0)(ES \neq 71)(ES_9 + S \neq 0)$

8) $¢\overline{ES}/G$ if $(\overline{ES_0})$ $(ES \neq 0)(ES \neq 71)(ES_9 + S \neq 0)$

At the end of 400 nanoseconds a \$ pulse will cause the following events:

1) A \$H to strobe the mantissa from AQ into H if $ES_9 + s = 0 + ES = 0$.

2) A \$N to strobe a mantissa into N if $/ES/<72 + ES = 0 + (S = 0)(ES_9)$.

3) A \$E to strobe the exponent from D into E if ES_9.

4) A \$G to strobe the proper count into G unconditionally.

5) LAFF to set if a maximum negative number was two's complemented and transferred to N.

6) K_{72} to reset if the two's complemented mantissa was transferred to N.

7) GE to reset.

8) GO to set if $ES = 0 + /ES/>71 + (\overline{ES_9})(S = 0)$.

9) GA to set if $(ES \neq 0)\ (/ES/<72)\ (ES_9 + S \neq 0)$.

10) \$ enters the timing pulse generator delay line network to produce another \$ pulse.

GA

During this cycle the mantissa in N is aligned so that its binary point falls at the same place as that of the mantissa in H. This is accomplished by shifting N right the number of places indicated by the G-counter. The control lines that become enabled to accomplish this are:

1) ¢N-16 (¢N_{0-55}/N_{16-71}, ¢N_0/N_{0-15}) if $(G > 14)(\overline{LAFF})$

2) ¢N-4(¢N_{0-67}/N_{4-71}, ¢N_0/N_{0-3}) if $(15 > G > 2)(\overline{LAFF})$

3) ¢N-1 (¢N_{0-70}/N_{1-71},¢N_0/N_0) if $(3 > G > 0)$ $(\overline{LAFF})$

4) ¢N-1 (¢N_{0-70}/N_{1-71},¢$\overline{N_0}$/N_0) if LAFF.

5) ¢N+1 (¢N_{1-71}/N_{0-70}) if REV.

6) LSA enabled if G = 1 + G = 4, + G = 16 + REV.

At the end of 230 nanoseconds a $ pulse will cause the following events:

1) A $N to strobe the shifted contents of N back into N.

2) A $G to decrement G by 16 if ¢N-16, by 4 if ¢N-4, by 1 if ¢N-1, or increment G by 1 if ¢N+1. or clear G if G = ±1, ±3, ±4, ±15, or ±16.

3) REV to set if (G = 15) + (G = 3).

4) GA to reset and GO to set if LSA enabled.

5) $ enters the timing pulse generator delay line network to produce another $ pulse.

GA will be repeated as many times as required to align the mantissa and will terminate on the first $ pulse following the enabling of LSA.

GO

During this cycle the complemented mantissa is added to the other mantissa in the S-adder producing the difference. This difference is stored in N, H is cleared to zero, and the proper indicators are set. The control line that becomes enabled to accomplish this is:

1) ¢S_{0-71}/N_{0-71}

At the end of 400 nanoseconds a $ pulse will cause the following events:

1) A $H to clear H to zero.

2) A $N to strobe the mantissa difference into N.

3) REV to set if an overflow occurred.

4) SIGN to set if the difference is negative and no overflow occurred.

5) ZERO to set if the difference is zero.

6) CARRY to set if there is a carry out of bit position 0.

7) CARRY to reset if there is no carry out of bit position 0.

8) K_{72} to reset.

9) GO to reset.

10) GN to set if an overflow occurred or if S_0 and S_1 are the same.

11) GF to set if no overflow occurred and $S_0 = S_1$.

12) $ enters the timing pulse generator delay line network to produce another $ pulse.

GN

During this cycle the mantissa in N is normalized. If an overflow occurred during GO the mantissa is shifted one place right with opposite sign inserted and the G-counter is decremented to (-1). In all other cases the mantissa is shifted left until N_0 and N_1 are not alike and the G-counter in incremented by 1 for each place that the mantissa is shifted. The control lines that become enabled to accomplish this are:

1) ¢N-1 (¢N_{0-71}/N_{1-71}, ¢$\overline{N_0}$/N_0) if REV set.

2) ¢N+16 (¢N_{16-71}/N_{0-55}) if N_0 through N_{15} are all the same.

3) ¢N+4 (¢N_{4-71}/N_{0-67}) if $\overline{\text{¢N+16}}$ and N_0 through N_3 are all the same.

4) ¢N+1 (¢N_{1-71}/N_{0-70}) if $(\overline{\text{¢N+16}})(\overline{\text{¢N+4}})$and $N_0 = N_1$.

5) LSN enabled if (¢N+16) $(N_{15} = N_{16} \neq N_{17})$ +(¢N+4) $(N_3 = N_4 \neq N_5)$ + (¢N+1) $(N_0 = N_1 \neq N_2)$ + (REV).

At the end of 230 nanoseconds a \$ pulse will cause the following events.

1) A \$N to strobe the shifted contents of N back into N.

2) A \$G to increment the G-counter by 16 if ¢N+16, by 4 if ¢N+4, by 1 if ¢N+1, or decrement G by 1 if REV is set.

3) REV to set if (¢N+16) $(N_{15} \neq N_{16})$ + (¢N+4) $(N_3 \neq N_4)$.

4) REV to reset if REV is set.

5) GN to reset and GF to set if LSN enabled.

6) \$ enters the timing pulse generator delay line network to produce another \$ pulse.

GN will be repeated as many times as required to normalize the mantissa and will terminate on the first \$ pulse following the enabling of LSN.

GF

During this cycle the mantissa in N is transferred to AQ, the count in G is subtracted from the exponent in E, and EOFL and EUFL are set if required. If the mantissa is zero the exponent will be forced to -128. The control lines that become enabled to accomplish this are:

1) ¢S_{0-71}/AQ_{0-71}.

2) ¢$\overline{G}$/ES and ¢CARRY/ES_7.

3) ¢ES/E if mantissa $\neq 0$.

4) ¢-2^7/E if mantissa = 0.

At the end of 400 nanoseconds a \$ pulse will cause the following events:

1) A \$AQ to strobe the mantissa into AQ.

2) A \$E to strobe the exponent into E.

3) EOFL to set if the subtraction of G from E caused the exponent to exceed +127.

4) EUFL to set if the subtraction of G from E caused the exponent to become smaller than -128.

5) GF to reset.

6) OUBF to reset if no other instruction has been initiated.

7) $ will generate a $LAST, which is sent to the CU to indicate completion of the DFSB instruction.

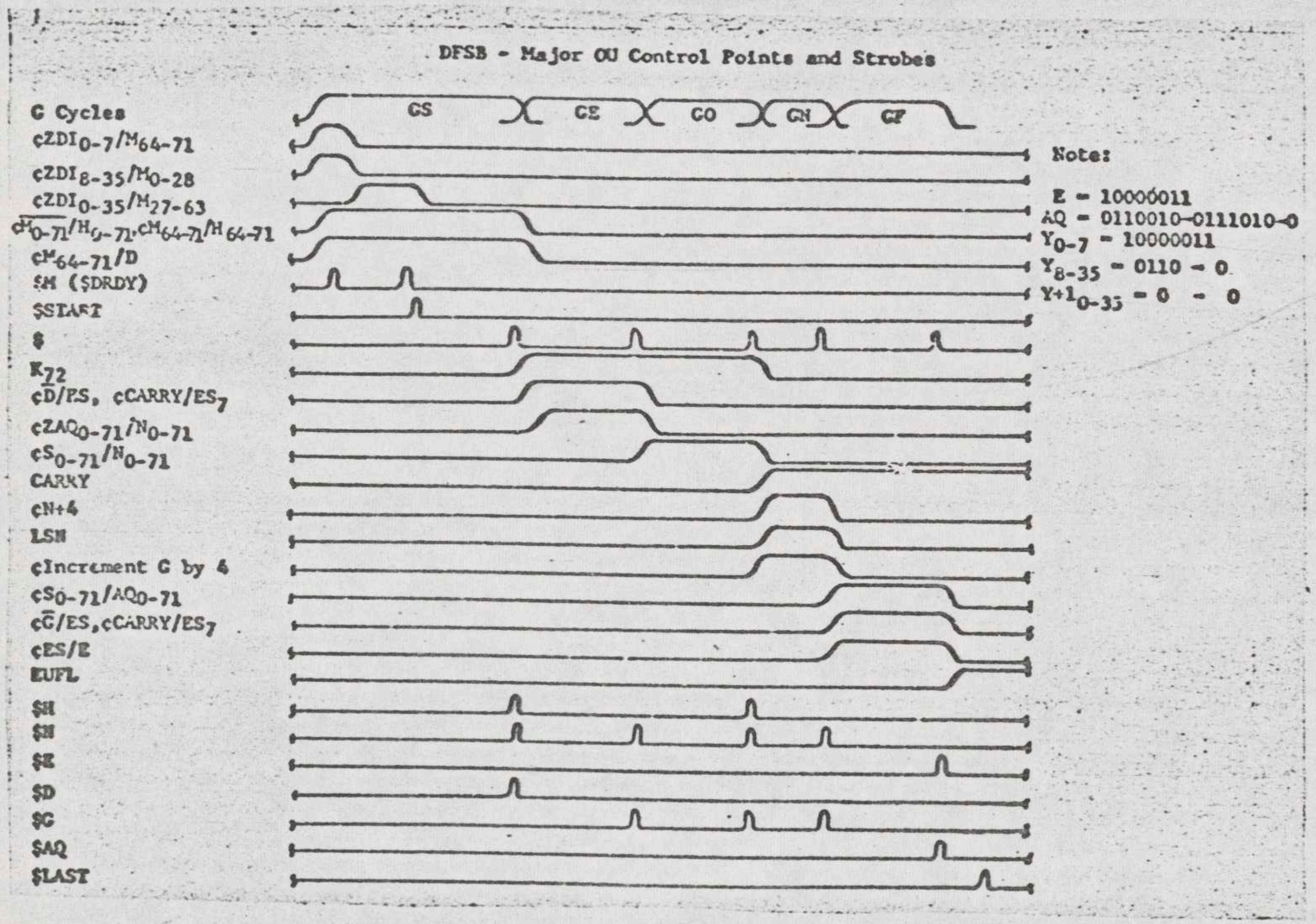

DFSB - Major OU Control Points and Strobes
G Cycles
GS
GE
GO
GN
GF
¢ZDI0-7/M64-71
¢ZDI8-35/M0-28
¢ZDI0-35/M27-63
¢M0-71/M0-71,¢M64-71/M64-71
¢M64-71/D
$M ($DRDY)
$START
$
K72
¢D/ES, ¢CARRY/ES7
¢ZAQ0-71/N0-71
¢S0-71/N0-71
CARRY
¢N+4
LSN
¢Increment G by 4
¢S0-71/AQ0-71
¢G/ES,¢CARRY/ES7
¢ES/E
EUFL
$H
$N
$E
$D
$G
$AQ
$LAST
Note:
E = 10000011
AQ = 0110010-0111010-0
Y0-7 = 10000011
Y8-35 = 0110 - 0
Y+1 0-35 = 0 - 0

fig 12.3.8.2

FIG. 12.3.8.2 DFSB Instruction, Timing Diagram

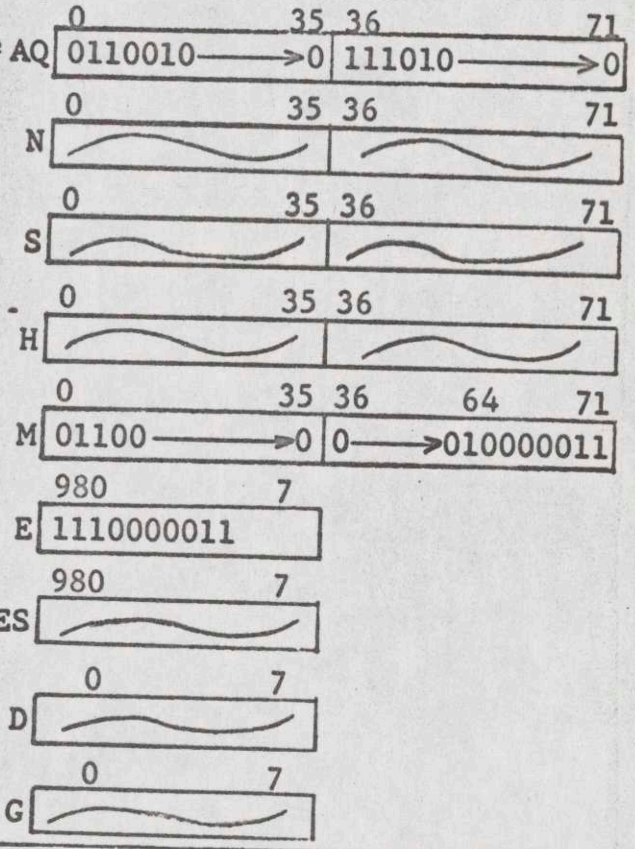
DFSB - Double Precision Floating Subtract Normalized

All registers are shown before the strobe at the end of the cycle

GS

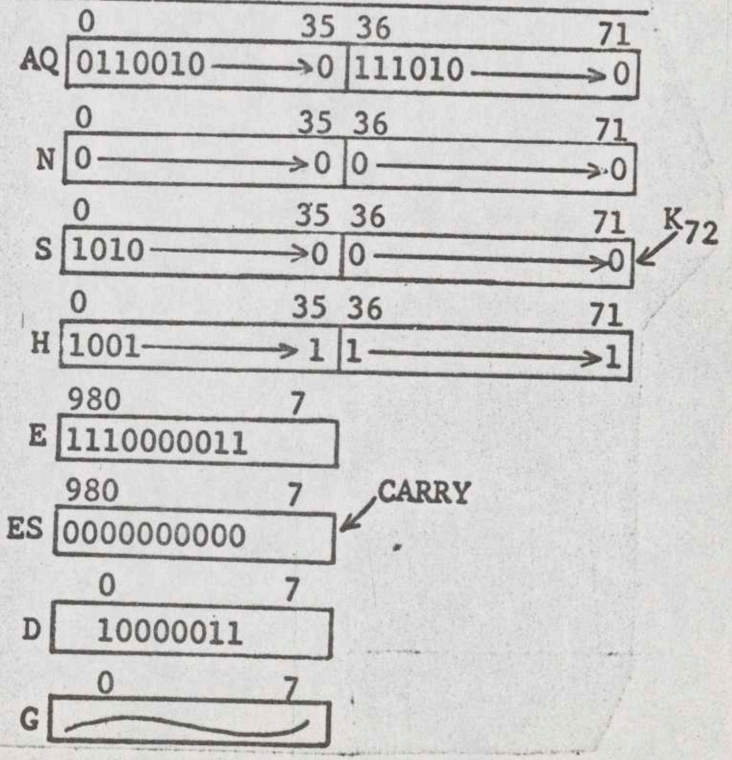
¢$\overline{M_{0-71}}/H_{0-71}$
¢M_{64-71}/H_{64-71}
¢M_{64-71}/D
¢Reset ZERO
¢Reset SIGN
¢Set K_{72}
\$H, \$N, \$D

} Strobe on \$


AQ
0
35 36
71
0110010→0
111010→0
N
0
35 36
71
S
0
35 36
71
H
0
35 36
71
M
0
35 36
64
71
01100→0
0→010000011
E
980
7
1110000011
ES
980
7
D
0
7
G
0
7

GE

¢AQ_{0-71}/ZAQ_{0-71}
¢ZAQ_{0-71}/H_{0-71}
¢D/E
¢$\overline{D}$/ES
¢$CARRY/ES_7$
¢ZAQ_{0-71}/N_{0-71} if ES = 0
\$N, \$G - Strobe on \$


AQ
0
35 36
71
0110010→0
111010→0
N
0
35 36
71
0→0
0→0
S
0
35 36
71
1010→0
0→0
K72
H
0
35 36
71
1001→1
1→1
E
980
7
1110000011
ES
980
7
0000000000
CARRY
D
0
7
10000011
G
0
7

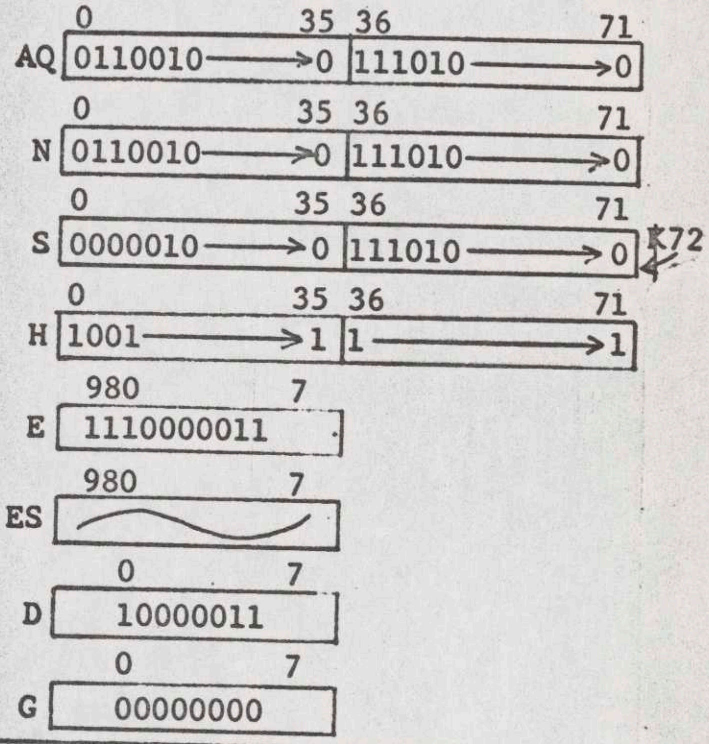
Note:
The next cycle will be a GO because ES = 0

FIG. 12.3.8.2a Register Contents during GS, GE Cycles of DFSB Instruction

GO

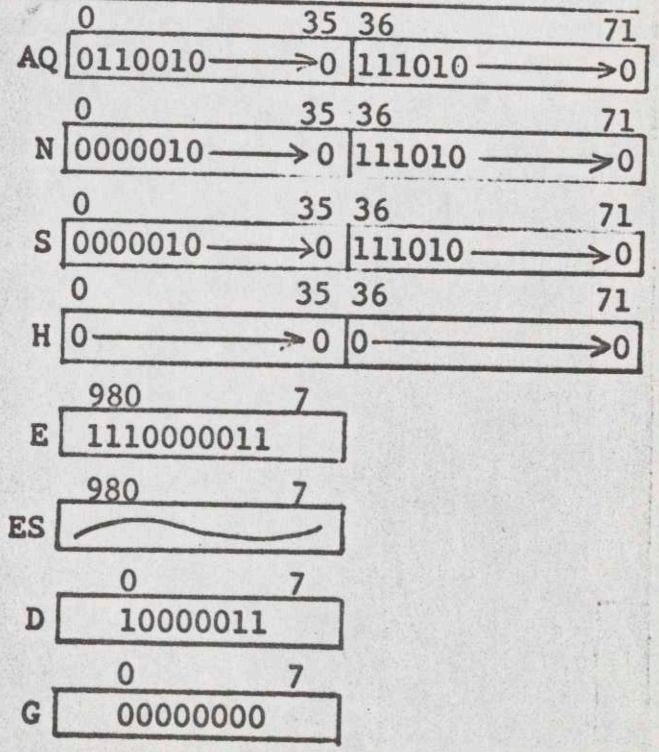
¢S_{0-71}/N_{0-71}
¢Set CARRY if carry out of S_0
¢Reset K_{72}
$H,$N,$G
} Strobe on $

CARRY will set


0 35 36 71
AQ 0110010→0 111010→0
0 35 36 71
N 0110010→0 111010→0
0 35 36 71
S 0000010→0 111010→0
K72
0 35 36 71
H 1001→1 1→1
980 7
E 1110000011
980 7
ES
0 7
D 10000011
0 7
G 00000000

GN

¢N+4(¢N_{4-71}/N_{0-67})
LSN
¢ Increment G by 4
$N,$G - Strobe on $


0 35 36 71
AQ 0110010→0 111010→0
0 35 36 71
N 0000010→0 111010→0
0 35 36 71
S 0000010→0 111010→0
0 35 36 71
H 0→0 0→0
980 7
E 1110000011
980 7
ES
0 7
D 10000011
0 7
G 00000000

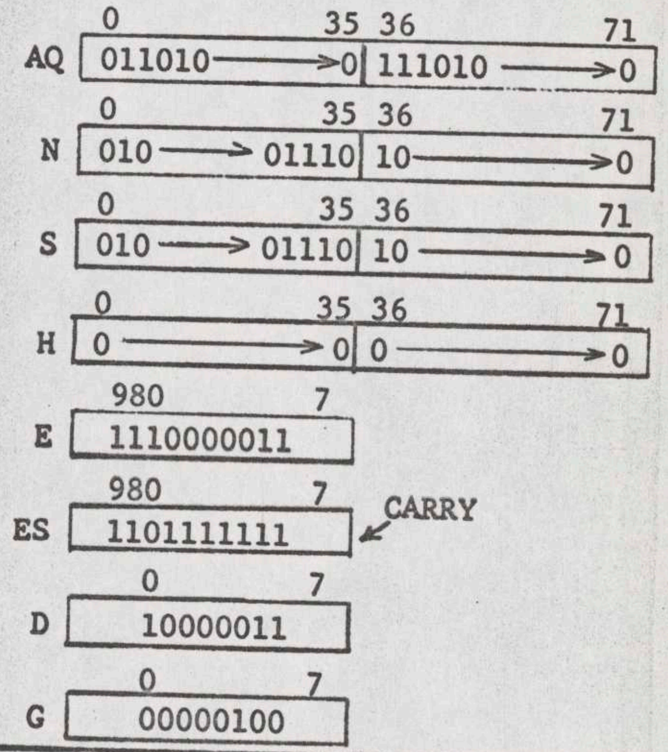
Note:
Last GN cycle

FIG. 12.3.8.2b Register Contents during GO, GN Cycles of DFSB Instruction

---

GF

$¢S_{0-71}/AQ_{0-71}$
$¢\bar{G}/ES$
$¢\ CARRY/ES_7$
¢ES/E
¢ Set EUFL
$E,$AQ } Strobe on $


AQ
0
35 36
71
011010 → 0
111010 → 0
N
010 → 01110
10 → 0
S
010 → 01110
10 → 0
H
0 → 0
0 → 0
E
980
7
1110000011
ES
1101111111
CARRY
D
0
10000011
G
00000100

Contents of the registers upon completion of the DFSB operation.

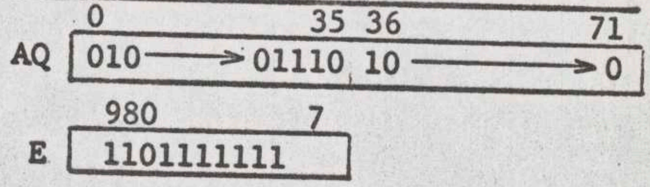

AQ
0
35 36
71
010 → 01110
10 → 0
E
980
7
1101111111

Fig 12.3.8.2c

FIG. 12.3.8.2c Register Contents during GF Cycle of DFSB Instruction

12.3.8.2-contd

Unnormalized

Under this heading fall the instructions DFCMG, DFCMP, DFLD, DUFA, and DUFS.

Example 1: Compare (DFCMP)

During this instruction the contents of memory locations Y and Y + 1 are compared with the contents of the Floating Register (EAQ) bits 0 through 63 and the Zero and Sign Indicators are set depending upon the results of the comparison. In general, this is accomplished by comparing the exponents of both numbers, selecting the number with the algebraically smaller exponent and shifting its mantissa right as many places as the difference of exponents states, then compare the mantissas, and set the indicators accordingly. The effective length of both mantissas is 64 bits (including sign). A compare is a subtract operation where the register contents are not altered. Since this operation is a straight compare the mantissa from memory is two's complemented and thus subtracted from the mantissa in AQ.

The data path from ZDI/M and the generation of $START are the same as for the DFSB instruction.

GS

During this cycle the complement of the mantissa in M is transferred to H, the exponent in M is transferred to D, and the proper indicator and control flip-flops are set up. The control lines that become enabled to accomplish this are:

1) ¢$\overline{M_{0-71}}/H_{0-71}$ and ¢M_{64-71}/H_{64-71}.

2) ¢M_{64-71}/D.

When the $START propagates the medium delay elements the operate strobe ($) occurs. This pulse causes the following events:

1) A $H to strobe the complement of the mantissa in M into H.

2) A $N to clear N to zero.

3) A $D to strobe the exponent in M into D.

4) ZERO and SIGN to reset.

5) K_{72} to set.

6) GS to reset and GE to set.

7) $ enters the timing pulse generator delay line network to produce another $ pulse.

GE

During this cycle the two exponents are compared by subtracting D from E, the difference if any is stored in G, the mantissa with the lesser exponent is stored in N, and the other mantissa is stored in H. The control lines that become enabled to accomplish this are:

1) ¢AQ_{0-71}/ZAQ_{0-71} unconditionally

2) ¢ZAQ_{0-71}/H_{0-71} unconditionally

3) ¢$\overline{D}$/ES and ¢CARRY/ES_7 unconditionally.

4) ¢ZAQ_{0-71}/N_{0-71} if $ES_9 + S = 0 + ES = 0$.

5) ¢S_{0-71}/N_{0-71} if $\overline{ES_9 + S = 0 + ES = 0}$.

6) ¢ES/G if $(ES_0)\ [(ES \neq 0)(ES \ngtr 71)(ES_9 + S \neq 0)]$

7) ¢$\overline{ES}$/G if $(\overline{ES_0})\ [(ES \neq 0)(ES \ngtr 71)(ES_9 + S \neq 0)]$

At the end of 400 nanoseconds a $ pulse will cause the following to happen:

1) A $H to strobe the mantissa from AQ into H if $\overline{ES_9 + S = 0 + ES = 0}$.

2) A $N to strobe a mantissa into N if $(/ES/ < 72) + (ES = 0) + (S = 0)\ (\overline{ES_9})$.

3) A $G to strobe the proper count into G.

4) LAFF to set if a maximum negative mantissa was two's complemented and transferred to N.

5) K_{72} to reset if the two's complemented mantissa was transferred to N.

6) GE to reset unconditionally.

7) GA to set if $(ES \neq 0)(/ES/ < 72) + (ES_9 + S \neq 0)$.

8) GF to set if $(ES = 0) + (/ES/ > 72) + (\overline{ES_9})(S = 0)$.

9) \$ enters the timing pulse generator delay line network to produce another \$ pulse.

GA

During this cycle the mantissa in N is aligned so that its binary point falls at the same place as that of the mantissa in H. This is accomplished by shifting N right the number of places indicated by the G-counter. The control lines that become enabled to accomplish this are:

1) ¢N-16 (¢N_{0-55}/N_{16-71}, ¢N_0/N_{0-15}) if $(G > 14)(\overline{LAFF})$

2) ¢N-4 (¢N_{0-67}/N_{4-71}, ¢N_0/N_{0-3}) if $(15 > G > 2)(\overline{LAFF})$

3) ¢N-1 (¢N_{0-70}/N_{1-71}, ¢N_0/N_0) if $(3 > G > 0)(\overline{LAFF})$.

4) ¢N-1 (¢N_{0-70}/N_{1-71}, ¢$\overline{N_0}$/N_0) if (LAFF).

5) ¢N+1 (¢N_{1-71}/N_{0-70}) if REV.

6) LSA enabled if $(G = 1) + (G = 4) + (G + 16)$ + (REV).

At the end of 230 nanoseconds a \$ pulse will cause the following events:

1) A \$N to strobe the shifted contents of N back into N.

2) A \$G to decrement G by 16 if ¢N-16, by 4 if ¢N-4, by 1 if ¢N-1, or increment G by 1 if ¢N+1, or clear G if LSA is enabled.

3) REV to set if (G = 15) + (G = 3).

4) LAFF to reset.

5) GA to reset and GF to set if LSA enabled.

6) $ enters the timing pulse generator delay line network to produce another $ pulse.

GA will be repeated as many times as required to align the mantissa, and will terminate on the first $ pulse following the enabling of LSA.

GF

During this cycle the complemented mantissa is added to the other mantissa in the S-adder producing the difference. This difference may cause an indicator to be set. If the mantissa in AQ is equal to the mantissa in Y and Y+1 the difference will be zero and ZERO will set. If the mantissa in AQ is greater than the mantissa in Y and Y+1 the difference will be positive and no indicators will set. If the mantissa in AQ is smaller than the mantissa in Y and Y+1 the difference will be negative and SIGN will set.

At the end of approxomately 400 nanoseconds a $ pulse will cause the following events:

1) SIGN to set if $(S_0)(\overline{\text{Overflow}})+(\overline{S_0})(\text{Overflow})$.

2) ZERO to set if (S = 0).

3) K_{72} to reset.

4) GF to reset.

$ will generate a $LAST, which is sent to the CU to indicate completion of the DFCMP instruction.

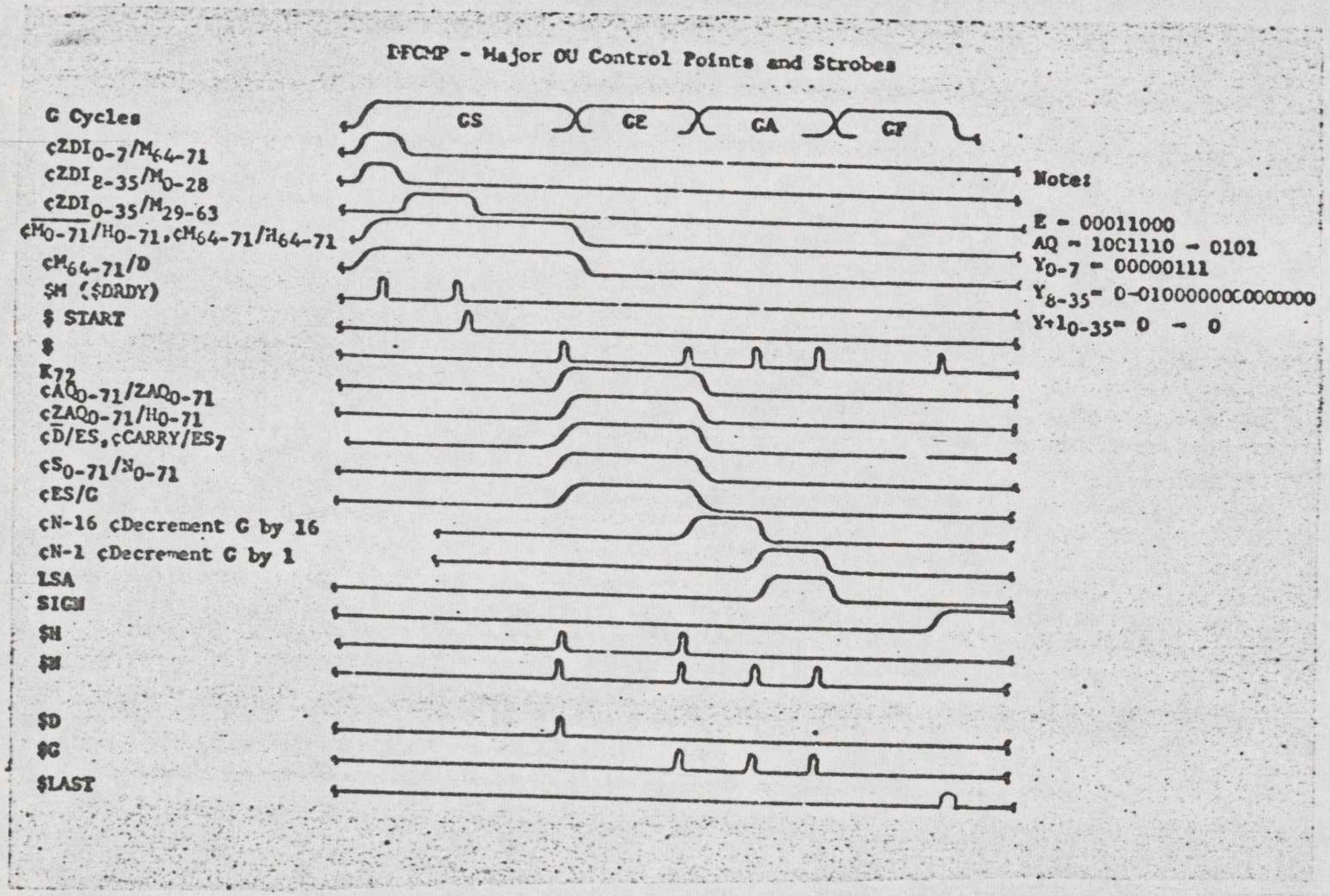

DFCMP - Major OU Control Points and Strobes
G Cycles
CS
CE
CA
CF
¢ZDI$_{0-7}$/M$_{64-71}$
¢ZDI$_{8-35}$/M$_{0-28}$
¢ZDI$_{0-35}$/M$_{29-63}$
¢M$_{0-71}$/H$_{0-71}$, ¢M$_{64-71}$/H$_{64-71}$
¢M$_{64-71}$/D
$M ($DRDY)
$ START
$
K72
¢AQ$_{0-71}$/ZAQ$_{0-71}$
¢ZAQ$_{0-71}$/H$_{0-71}$
¢$\overline{D}$/ES, ¢CARRY/ES$_7$
¢S$_{0-71}$/N$_{0-71}$
¢ES/G
¢N-16 ¢Decrement G by 16
¢N-1 ¢Decrement G by 1
LSA
SIGN
$H
$N
$D
$G
$LAST
Note:
E = 00011000
AQ = 10C1110 - 0101
Y$_{0-7}$ = 00000111
Y$_{8-35}$ = 0-01000000C0000000
Y+1$_{0-35}$ = 0 - 0

FIG. 12.3.8.2d DFCMP Instruction, Timing Diagram

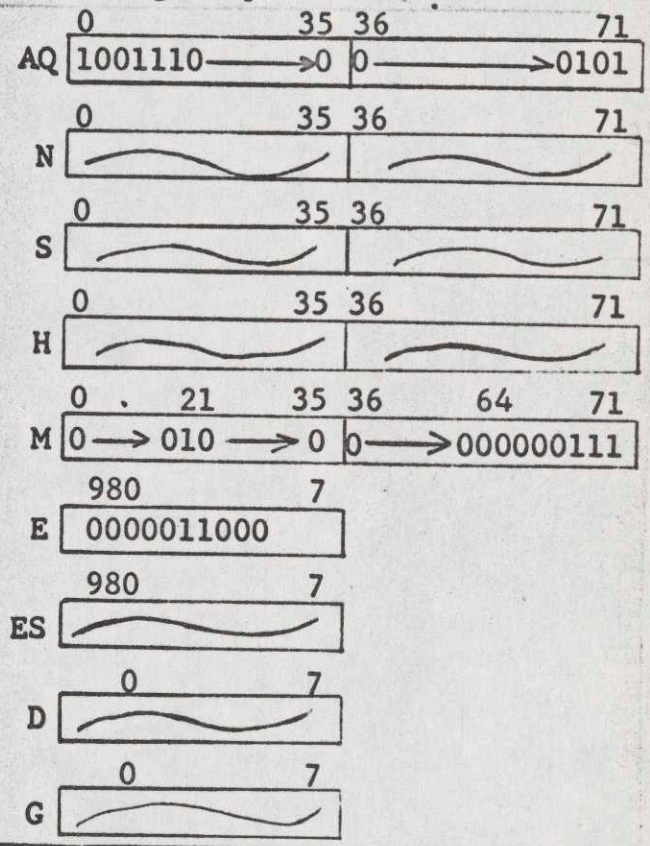
DFCMP - Double Precision Floating Compare

All registers are shown before the strobe at the end of the cycle.

GS

¢$\overline{M_{0-71}}$/H_{0-71}, ¢M_{64-71}/H_{64-71}
¢M_{64-71}/D
¢ Set K_{72}
¢ Reset ZERO
¢ Reset SIGN
\$H, \$N, \$D

Strobe on \$


AQ
0
35 36
71
1001110 → 0
0 → 0101
N
S
H
M
0 21 35 36 64 71
0 → 010 → 0
0 → 000000111
E
980 7
0000011000
ES
D
0 7
G

GE

¢AQ_{0-71}/ZAQ_{0-71}
¢ZAQ_{0-71}/H_{0-71}
¢$\overline{D}$/ES and ¢CARRY/ES_7
¢S_{0-71}/N_{0-71}
¢ES/G
¢ Reset K_{72}
\$H, \$N, \$G

Strobe on \$

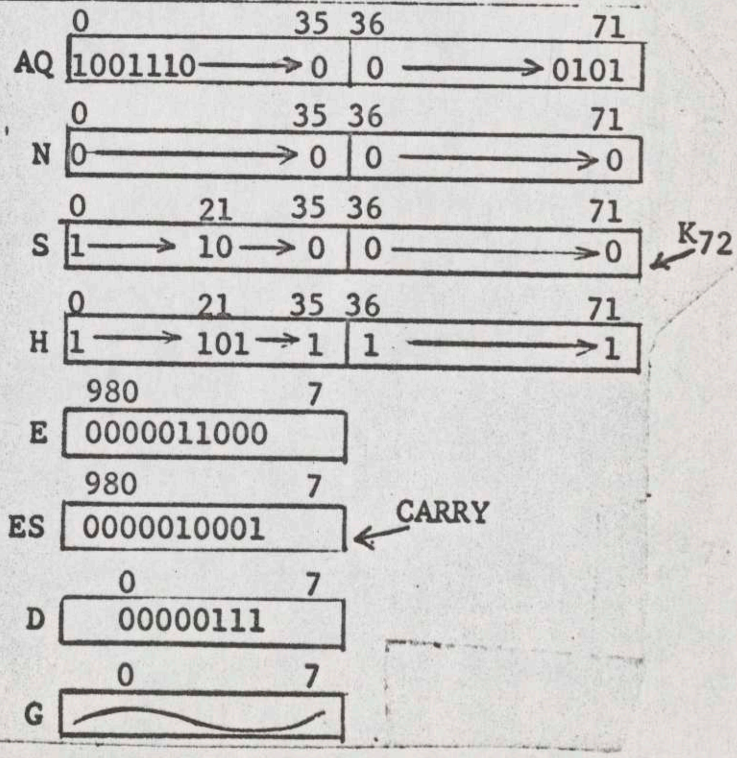
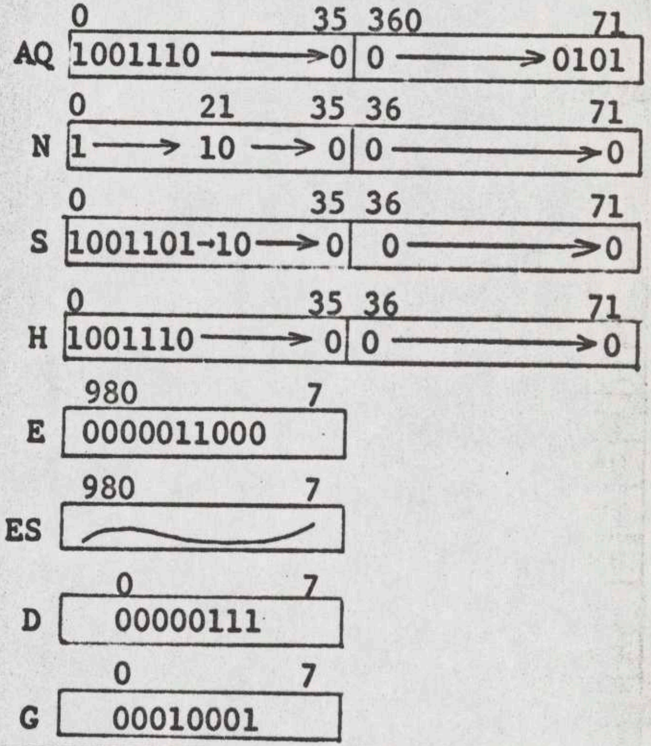

AQ
1001110 → 0
0 → 0101
N
0 → 0
0 → 0
S
1 → 10 → 0
0 → 0
K_{72}
H
1 → 101 → 1
1 → 1
E
980 7
0000011000
ES
0000010001
CARRY
D
0 7
00000111
G

FIG. 12.3.8.2e Register Contents during GS, GE Cycles of DFCMP Instruction

GA (first)

¢N-16
¢ Decrement G by 16 } Strobe on $
$N, $G


AQ
0 35 360 71
1001110 → 0 0 → 0101
N
0 21 35 36 71
1 → 10 → 0 0 → 0
S
0 35 36 71
1001101→10 → 0 0 → 0
H
0 35 36 71
1001110 → 0 0 → 0
E
980 7
0000011000
ES
980 7
D
0 7
00000111
G
0 7
00010001

GA (second)

¢N-1
LSA
¢ Decrement G by 1 } Strobe on $
$N, $G

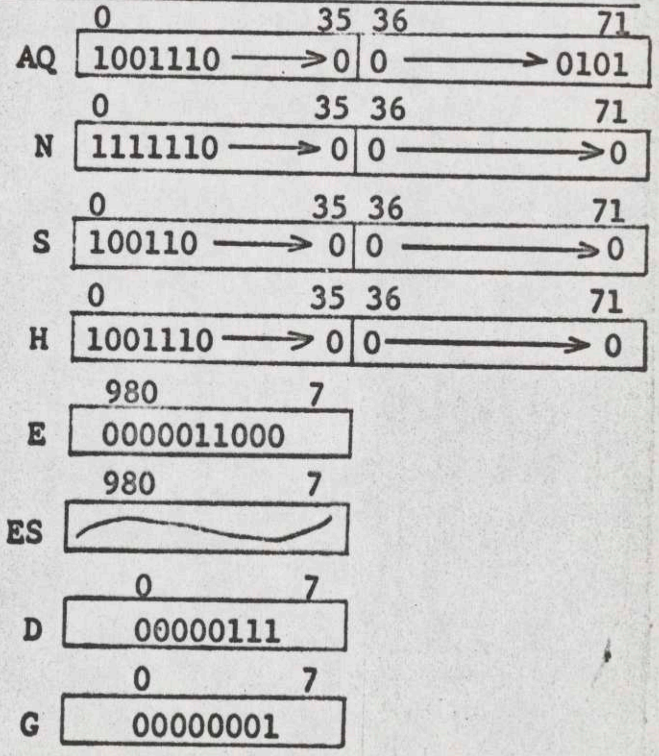

AQ
0 35 36 71
1001110 → 0 0 → 0101
N
0 35 36 71
1111110 → 0 0 → 0
S
0 35 36 71
100110 → 0 0 → 0
H
0 35 36 71
1001110 → 0 0 → 0
E
980 7
0000011000
ES
980 7
D
0 7
00000111
G
0 7
00000001

FIG. 12.3.8.2f Register Contents during GA Cycle of DFCMP Instruction

GF

¢ SET SIGN - Strobe on $

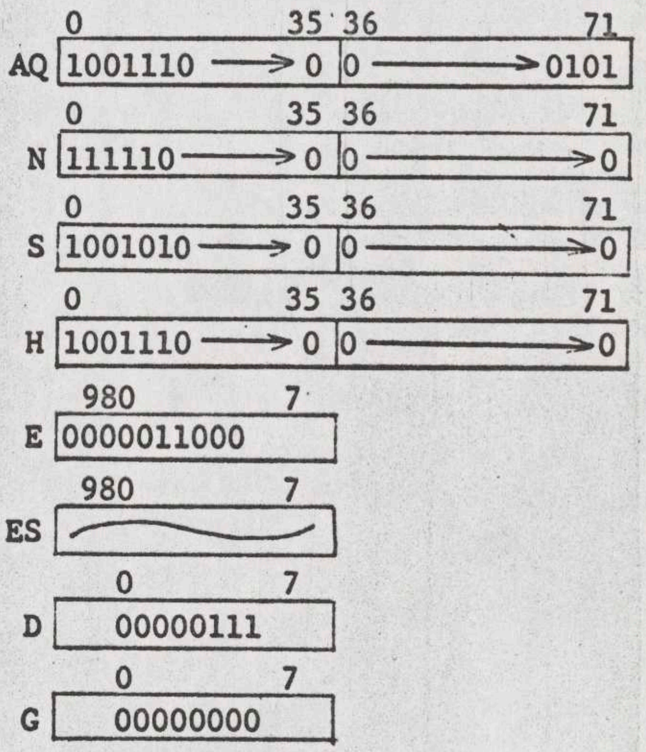

0 35 36 71
AQ 1001110 → 0 0 → 0101
0 35 36 71
N 111110 → 0 0 → 0
0 35 36 71
S 1001010 → 0 0 → 0
0 35 36 71
H 1001110 → 0 0 → 0
980 7
E 0000011000
980 7
ES
0 7
D 00000111
0 7
G 00000000

Fig 12.3.8.2g

FIG. 12.3.8.2g Register Contents during GF Cycle of DFCMP Instruction

12.3.9 Floating Point Store Group

This group consists of the three floating point store instructions FST, DFST, and STE. The major data paths and control points for these instructions are shown in Fig. 12.3.9.

To illustrate the control used in the execution of these instructions two instructions will be explained in detail: For the detailed control of all instructions in this group, refer to OU Specification Chart 43D140232, page 6.

In the explanation of all the example instructions, the following conditions will be assumed.

1) The instruction has been strobed into the O-register.

2) The GS and GC cycle flip-flops have been set.

3) The DDF flip-flop has been set indicating the operation decode has settled.

NOTE:

Reference to Fig. 12.3.9 will aid in the understanding of the instruction execution and implementation.

12.3.9.1 Single Precision

The instructions comprising the single precision portion of the Floating Point Store Group are FST and STE.

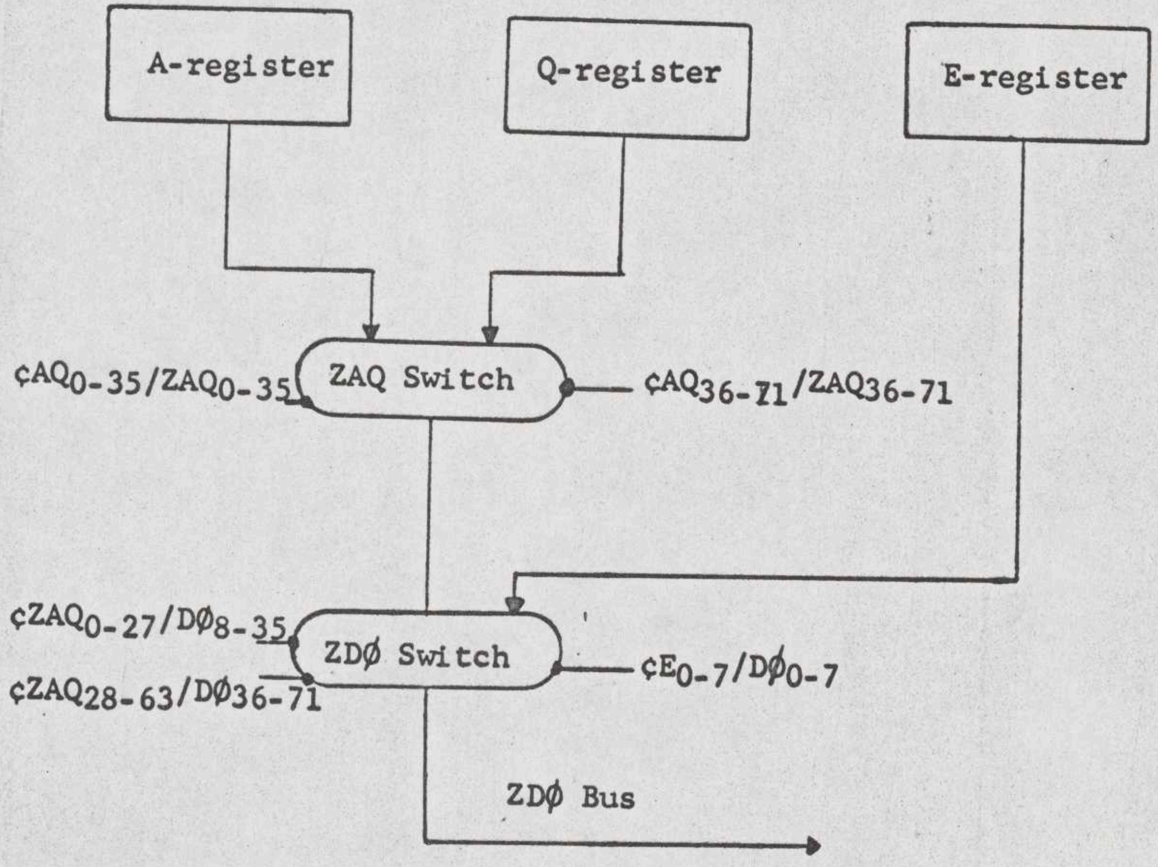

A-register
Q-register
E-register
¢AQ$_{0-35}$/ZAQ$_{0-35}$
ZAQ Switch
¢AQ$_{36-71}$/ZAQ$_{36-71}$
¢ZAQ$_{0-27}$/DØ$_{8-35}$
¢ZAQ$_{28-63}$/DØ$_{36-71}$
ZDØ Switch
¢E$_{0-7}$/DØ$_{0-7}$
ZDØ Bus

Fig. 12.3.9
Floating Point Store Group, Major Flowchart

12.3.9.1 - contd

Example 1: Floating Store (FST)

During this instruction the contents of bits 0 through 7 of the E-register are transferred through the ZDO switch to bits 0 through 7 of a memory location Y and the contents of bits 0 through 27 of the A-register are transferred through the ZAQ and ZDO switches to bits 8 through 35 of the same memory locstion Y.

The ZAQ and ZDO switches are enabled by the presence of the FST decode together with the GS flip-flop set and the control point ¢YS/ZDO not enabled. The GC flip-flop will remain set from the beginning of the GS cycle to the end of the GF cycle. The ZAQ and ZDO switches are enabled by control points ¢AQ$_{0-35}$/ZAQ$_{0-35}$, ¢ZAQ$_{0-27}$/ZDO$_{8-35}$ and ¢E$_{0-7}$/ZDO$_{0-7}$.

When the $ Set GS pulse was generated for this instruction, it was delayed by about 40 nanoseconds to become $YRY and it becomes $SDY (store data ready). $YRY sets the operand ready flip-flop, and is returned to the CU to signal that a new operation code may be sent to the OU. YRY set allows the $ START pulse to be generated and enter the Timing Pulse Generator to later produce the first operate strobe $. $ START sets the OU busy flip-flop, OUB. $SDY is sent to the CU to initiate the memory store cycle prior to the occurrence of the first $ strobe in the OU.

When the $ START pulse propogates the medium delay elements the first operate strobe $ occurs. This causes the following events:

1) The GS cycle to reset and the GF cycle to set.

2) The O-register Busy flip-flop (ORB) to reset.

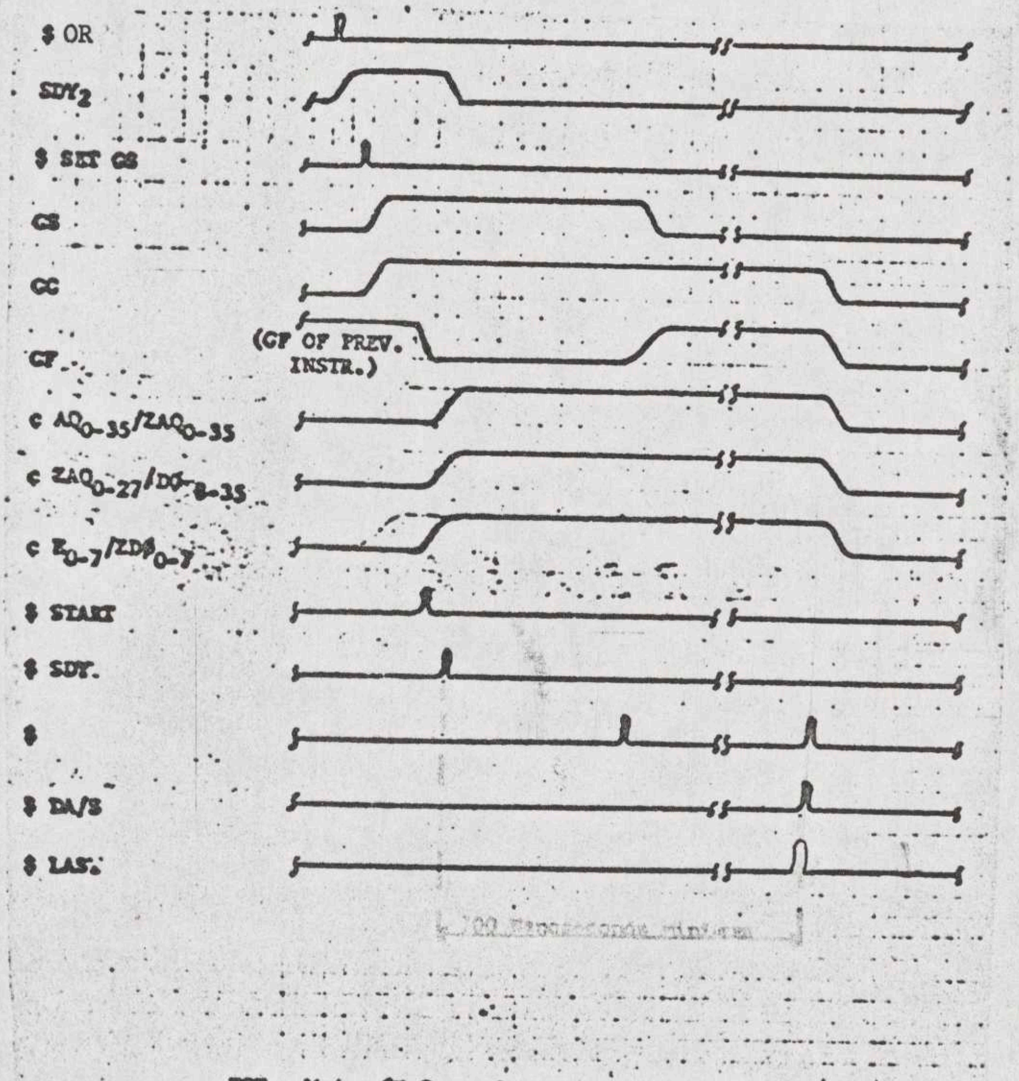
3) The $ strobe re-enters the Timing Pulse Generator, but it is inhibited from exiting the delay line network.

The operand will be stored in the Data Out Buffer, in the Interface Frame, by $INT. $INT will also generate a $DS and send it to the OU. $DS is "OR3d" into the Timing Pulse Generator and exits on the general strobe ($) lines. This pulse causes:

1) GF and GC to reset.

2) OUB to reset if no other instruction has been initiated.

3) It will generate a $ LAST, which is sent to the CU to indicate completion of the FST instruction.


$ OR
SDY2
$ SET GS
GS
GC
GF
(GF OF PREV. INSTR.)
c AQ0-35/ZAQ0-35
c ZAQ0-27/DG8-35
c E0-7/ZDØ0-7
$ START
$ SDY.
$
$ DA/S
$ LAS.

FST - Major ØU Control Points and Strobes

Fig 12.3.9.1

FIG. 12.3.9.1 FST Instruction, Timing Diagram

12.3.9.2 Double Precision

The only instruction in the double precision portion of the Floating Point Store Group is DFST.

Example 1: Double Precision Floating Store (DFST)

During this instruction the contents of bits 0 through 7 of the E-register are transferred through the ZDO switch to bits 0 through 7 of a memory location Y and the contents of bits 0 through 63 of the AQ-register are transferred through the ZAQ and ZDO switches to bits 8 through 35 of Y and bits 0 through 35 of Y + 1.

GS

The ZAQ and ZDO switches are enabled by the presence of the DFST decode together with the GC flip-flop set and the control point ¢YS/ZDO not enabled. The GC flip-flop will remain set for two cycles. The ZAQ and ZDO switches are enabled by ¢AQ$_{0-71}$/ZAQ$_{0-71}$, ¢ZAQ$_{0-27}$/ZDO$_{8-35}$, and ¢E$_{0-7}$/ZDO$_{0-7}$.

When the $ Set GS pulse was generated for this instruction, it was delayed by about 40 nanoseconds to become $YRY and it becomes $SDY. $YRY sets the operand ready flip-flop YRY, and is returned to the CU. YRY set allows the $ START pulse to be generated and enter the OU Timing Pulse Generator to later produce the operate strobe $. $ START sets the OU busy flip-flop OUB.

$SDY is sent to the CU to initiate the store cycle prior to the occurrence of the first $ strobe in the OU.

GS

When the $ START pulse propogates the medium delay elements the operate strobe $ occurs. This causes the following events:

1) The GS cycle to reset and the GT cycle to set.

2) The O-register busy flip-flop ORB, to reset.

3) A level consisting of DP Stores and GT goes high.

4) The $ strobe re-enters the Timing Pulse Generator, but since a store is being executed it is inhibited from exiting onto the general $ lines.

GT

After a minimum time of 300 nanoseconds after $SDY, the first half of the store operation will be complete and the first $DS will be received from the Interface Frame to signal this fact. $DS will produce a $ strobe which will:

1) Reset the GC and GT cycle flip-flops.

2) Set the GF and GFT cycle flip-flops.

3) Cause the DP Stores and GT level to go low after about 60 nanoseconds but only after $ strobe has "ANDed" with DPA to produce $DP which is delayed by approximately 270 nanoseconds before being sent to the Interface Frame to signify that the second data word is ready to be stored.

GF

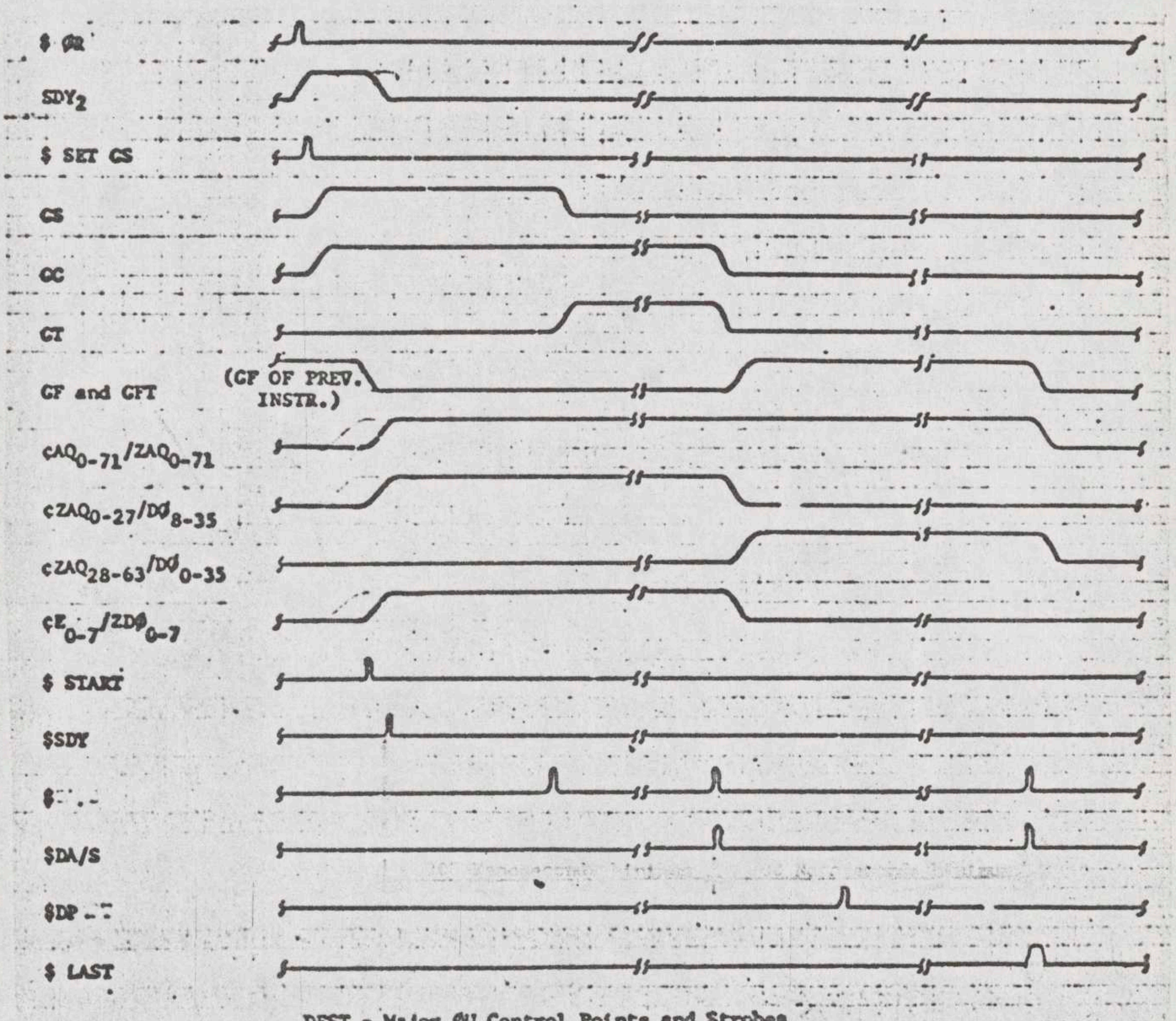
During this cycle the last half of the double precision operand is stored in memory location Y + 1. The ZAQ and ZDO switches are enabled by ¢AQ_{0-71}/ZAQ_{0-71} and ¢ZAQ_{28-63}/ZDO_{0-35}. This allows the second half of the operand to be present on the ZDO lines before the OU send the $DP pulse to the Interface Frame for the second half store.

The $DP strobes the data into the Data Out Buffer bits 36-71 and is returned as the second $DS indicating the second half data has been received. This $DS will generate a $ Strobe pulse which will:

1) Rest the GF and GFT cycles.

2) Reset the OUB flip-flop if no other instruction has been initiated.

3) Generate $ LAST, which is sent to the CU to indicate completion of the DFST instruction.


$ GR
SDY_2
$ SET CS
CS
GC
GT
GF and GFT
(GF OF PREV. INSTR.)
¢AQ_{0-71}/ZAQ_{0-71}
¢ZAQ_{0-27}/$DØ_{8-35}$
¢ZAQ_{28-63}/$DØ_{0-35}$
¢E_{0-7}/$ZDØ_{0-7}$
$ START
$SDY
$DA/S
$DP
$ LAST

DFST - Major ØU Control Points and Strobes

Fig 12.3.9.2

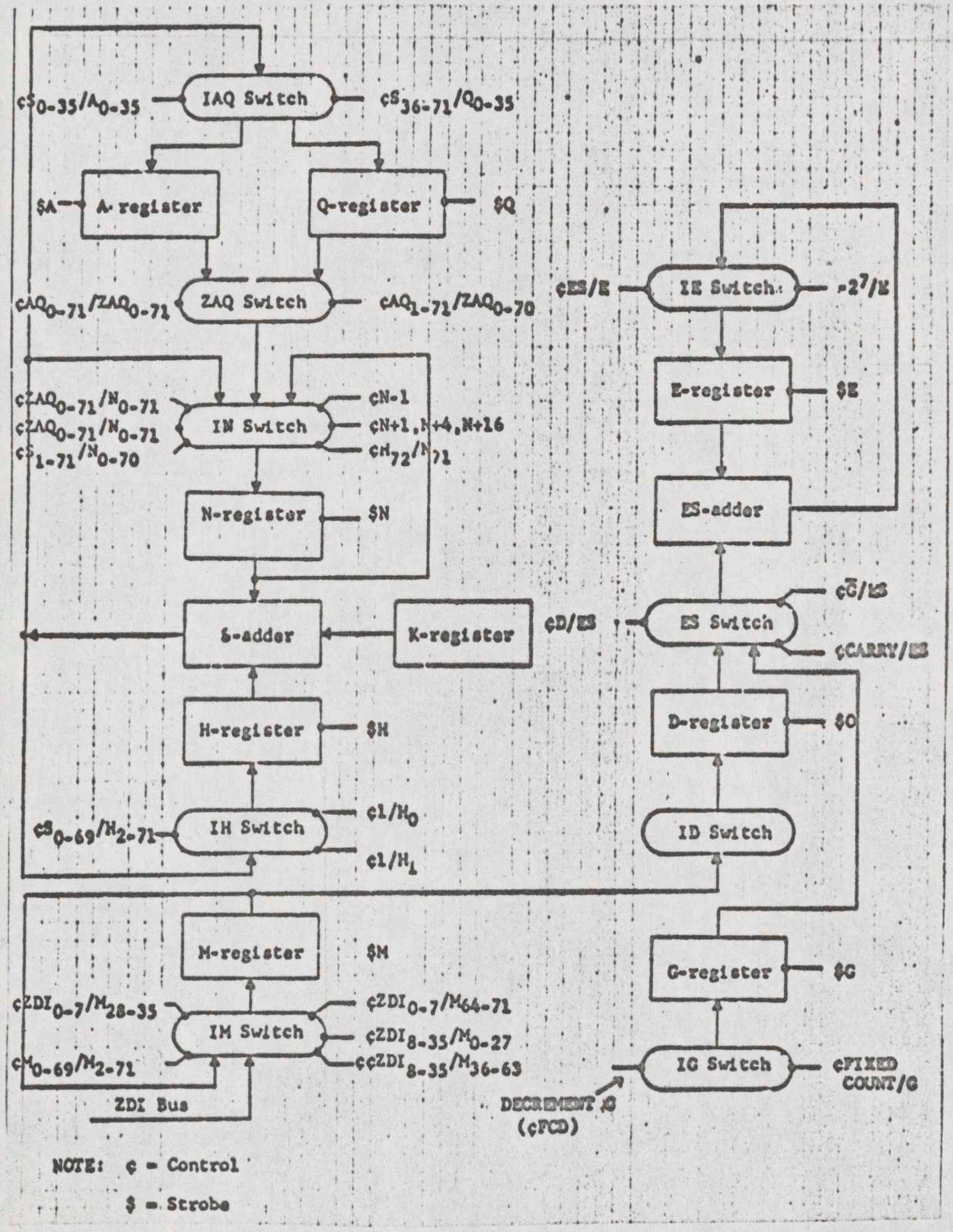
FIG. 12.3.9.2 DFST Instruction, Timing Diagram

12.3.10 Floating Point Multiply Group

This group of four instructions provides the floating point multiply capabilities of this machine. The major data paths and control points for these instructions are shown in Fig. 12.3.10.

12.3.10.1 Single Precision

The instructions FMP and UFM provide single precision floating multiplication capability. For both instructions a 72-bit mantissa (multiplicand) is multiplied by a 28-bit mantissa (multiplier) to produce a 72-bit product. If a Direct Upper (DU) modifier is specified the multiplier mantissa consists of $c(Y_{8-17})$ followed by 8 zeros. If a Direct Lower (DL) modifier is specified the multiplier mantissa consists of 10 leading zeros followed by $c(Y_{18-35})$. The exponent for a DL multiplier is zero. In all cases the multiplier mantissa is contained in the M-register, right justified to bit 63, the multiplicand mantissa is contained in the AQ-register, the partial products are accumulated in the H-register, and the final mantissa product is stored in the AQ-register. The multiplier exponent is transferred from M_{64-71} to D, the multiplicand exponent is contained in E, and the exponent sum is stored in E.


IAQ Switch
¢S_{0-35}/A_{0-35}
¢S_{36-71}/Q_{0-35}
$A
A-register
Q-register
$Q
ZAQ Switch
¢AQ_{0-71}/ZAQ_{0-71}
¢AQ_{1-71}/ZAQ_{0-70}
IN Switch
¢ZAQ_{0-71}/N_{0-71}
¢ZAQ_{0-71}/N_{0-71}
¢S_{1-71}/N_{0-70}
¢N-1
¢N+1, N+4, N+16
¢H_{72}/N_{71}
N-register
$N
δ-adder
K-register
H-register
$H
IH Switch
¢S_{0-69}/H_{2-71}
¢$1/H_0$
¢$1/H_1$
M-register
$M
IM Switch
¢ZDI_{0-7}/M_{28-35}
¢M_{0-69}/M_{2-71}
¢ZDI_{0-7}/M_{64-71}
¢ZDI_{8-35}/M_{0-27}
¢¢ZDI_{8-35}/M_{36-63}
ZDI Bus
IE Switch
¢ES/E
$-2^7/E$
E-register
$E
ES-adder
ES Switch
¢D/ES
¢$\overline{C}$/ES
¢CARRY/ES
D-register
$D
ID Switch
G-register
$G
IG Switch
¢FIXED COUNT/G
DECREMENT G
(¢FCD)

NOTE: ¢ = Control
$ = Strobe

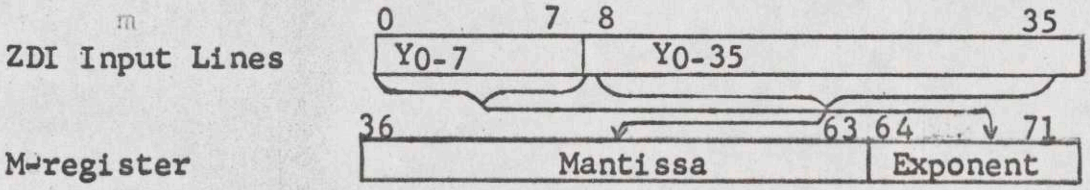
Fig. 12.3.10

Floating Point Divide Group, Major Flowchart

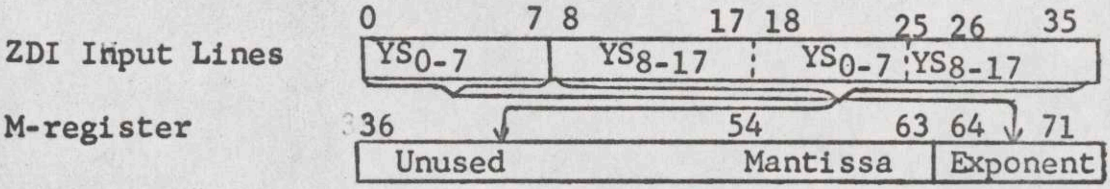
12.3.10.1 - contd

The possible multiplier configurations which may be contained in the M-register with respect to the ZDI lines from the CU are shown below:

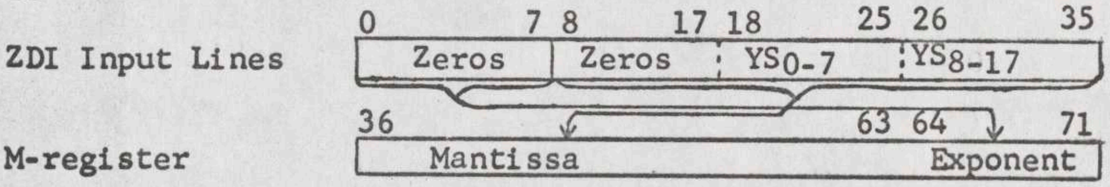
1) Normal


0
7
8
35
ZDI Input Lines
Y_{0-7}
Y_{0-35}
36
63
64
71
M-register
Mantissa
Exponent

2) DU


0
7
8
17
18
25
26
35
ZDI Input Lines
YS_{0-7}
YS_{8-17}
YS_{0-7}
YS_{8-17}
M-register
36
54
63
64
71
Unused
Mantissa
Exponent

3) DL


0
7
8
17
18
25
26
35
ZDI Input Lines
Zeros
Zeros
YS_{0-7}
YS_{8-17}
36
63
64
71
M-register
Mantissa
Exponent

YS_{0-17} is the Address Register output through the YS-adder.

The only difference between a FMP and an UFM instruction is in the handling of the mantissa product. During a FMP the mantissa product is always normalized, while during an UFM the mantissa product is normalized only if an overflow occurred.

For the detailed control of these two instructions refer to the OU Specification Chart 43D140232, page 8.

NOTE: Reference to Fig. 12.3.10 will aid in the understanding of the instruction execution and implementation.

12.3.10.1- contd

Example 1: Floating Multiply Normalized (FMP)

During this instruction the contents of the AQ-register are multiplied by the contents of memory location Y_{8-35} to produce a 72-bit (including sign) product which is stored in the AQ-register, and the contents of the E-register are added to the contents of memory location Y_{0-7} to produce an 8-bit sum which is stored in the E-register. For this example no operands are used, but all major controls are explained.

The IM-switch is enabled by the presence of the FMP decode together with DIC and DPFF reset. The control points ¢ZDI_{0-7}/M_{64-71}, and ¢ZDI_{8-35}/M_{36-63} are enabled to allow the ZDI bus to be present at the M-register inputs. When the operand ready strobe $DRDY is received from the CU, DPFF reset allows the M-register strobe $\$M_{0-71}$ to be generated and strobe the multiplier into the M-register. See page 12.232 for the possible arrangements of the multiplier. If the multiplier operand being furnished by the CU is a Direct Upper, a $DMF will be received at the same time as $DRDY. $DMF will set DUM. The $DRDY pulse is turned around as $YRY and returned to the CU to signal that a new instruction may be prepared. $YRY also sets YRY, and because of the FMP decode sets DIC. The DIC flip-flop is used as an inhibit on the ZDI/M controls for the duration in which the multiplier is contained in M. When YRY is set, the conditions exist to allow a $ START to be generated by a last of three pulse detector. $ START sets the OU Busy flip-flop (OUB), resets YRY and DDF, and transfers the operation code from the O-register to the C-register. It also enters the timing pulse generator delay line network.

12.3.10.1 contd

GS

During the 320 nanoseconds of GS from \$ START to the first \$, control is established to load the G-counter with the proper count with the first \$. If DUM is set (which defines the multiplier is a Direct Upper) Fixed Count/G control will force a count of 6 into G* (that is, set G_5 and G_6). If DUM is reset, a count of 15 is placed in G (that is, set G_4, G_5, G_6, and G_7). Also the control line ¢M_{64-71}/D becomes enabled to transfer the exponent to the D-register.

As per the multiply algorithm, the low-order two bits of the multiplier M_{62-63} are decoded during this same 320 nanoseconds of GS to indicate whether the first multiplication will be by 0, 1, 2, or 3. The decode of M_{62-63} is interpreted as:

1) If M_{62-63} = 00, this represents multiplication by zero or a pass condition (that is, no modification of the partial product in H which at this time is zero). No ZAQ or IN switch is enabled.

2) If M_{62-63} = 01, this represents multiplication of the multiplicand by one. Since at this time there is no partial product. To accomplish this the ZAQ and IN switches are enabled by ¢AQ_{0-71}/ZAQ_{0-71} and ¢ZAQ0-71/N0-71. These enabled switches allow the A-register multiplicand to be present at the N-register inputs.

---

*Refer to paragraph 12.4 for a description of the G-counter when used as a decrement counter.

12.3.10.1-contd

3) If $M_{62-63} = 10$, this represents multiplication of the multiplicand by two. For a binary number this may be done by shifting the number left by one bit position. To accomplish this the ZAQ and IN switches are enabled by ¢AQ_{1-71}/ZAQ_{0-70} and ¢ZAQ_{0-71}/N_{0-71}.

4) If $M_{62-63} = 11$, this represents multiplication of the multiplicand by three. This is accomplished by doing a multiply of (4-1 = 3). The multiply by 4 is accomplished by adding one to the bit which will be contained in M_{63} during the next multiply cycle. Effectively this means that the bits presently contained in M_{60-61} are to be incremented by one. The minus one control is set up during the present multiply cycle and is performed by subtracting the contents of A from the partial product. The IAQ and IN switches are enabled by ¢AQ_{0-71}/ZAQ_{0-71}, ¢$\overline{ZAQ}_{0-71}$/N_{0-71}. These switches present the one's complement of the AQ-register to N_{0-71}.

Because a new set of multiplier bits will be needed for the next multiply cycle (the first GO cycle), the control is set up to allow a right shift of the M-register by two. The IM switch is enabled by ¢M_{0-33}/M_{2-35} and ¢M_{34-69}/M_{36-71} to accomplish this right shift.

When the $ START pulse propogates the medium delay elements, the first operate strobe $ occurs. This allows the following events:

1) The GS cycle to reset and the GO cycle to set.

2) The ZERO and SIGN Indicators to reset.

11) $ enters the timing pulse generator delay line network to produce another $ pulse.

(GO) (G > 3)

During this 320-nanosecond time the first GO cycle is true and the S-adder is used to produce the first partial product and make it available on the S bus at the adder output. While this partial product is being developed, the control points are established to store the partial product on the S bus into H and set up the control to perform the next 2-bit multiplication.

During the GO cycles the determination of what the next effective multiplier will be is not only a function of M_{62-63} as it was in GS, but it is also dependent on the state of the String flip-flop (STFF). Although there are still only four possibilities for a multiplier, there are now two conditions that cause each possibility. These are shown below in truth table form:

M_{62}	M_{63}	STFF	Effective Multiplier
0	0	0	0
0	1	0	+1
1	0	0	+2
1	1	0	3 (or -1)
0	0	1	+1
0	1	1	+2
1	0	1	3 (or -1)
1	1	1	0

The control points enabled as a function of the effective multiplier are the same as the ones mentioned during GS when the multiplier was 0, 1, 2, or 3.

The IH switch is enabled during GO to allow the partial product to be strobed into H. The control points enabled for this during GO are ¢S_{0-33}/H_{2-35} and ¢S_{34-69}/H_{36-71}.

During this GO cycle the controls on the IM switch are again enabled to allow the M-register to be right shifted by two in order to have a new set of multiplier bits for the next cycle. The Full Counter Decrement control (¢FCD) becomes enabled so that the next $G will decrement the count by one.

When the first $ strobe propogates the medium delay elements, the second $ is distributed to the OU to end the first GO cycle and the following events occur:

1) The GO flip-flop is strobed, but because the G count is not zero it remains set.

2) $G decrements the G-counter by one.

3) $M strobes the M-register and executes a right shift of two to set up a new set of multiplier bits.

4) $H strobes the partial product into H_{2-71}. The content of the Partial Product Sign flip-flop (PPS) is strobed into H_0 unconditionally and into H_1 if no overflow occurred.

5) $N strobes the new premultiplied multiplicand in the N-register as determined by the multiplier decode.

6) The K_{72} adder control flip-flop and the String flip-flop (STFF) are set if the effective multiplier is 3. STFF is reset if the effective multiplier is +1, +2, or ±2 and not changed if it is zero. K_{72} is reset for values of other than 3.

7) The PASS flip-flop is set for an effective multiplier of zero, or reset for 1, 2, or 3.

8) The PPS flip-flop is strobed and sets or resets for the same conditions listed in GS.

9) \$ enters the Timing Pulse Generator delay line network to produce another \$ pulse.

Up until the time when the G-counter is decremented to a count of 3 the functions performed during all of the GO cycles are iterative of the first GO cycle.

GO (G = 3)

During the GO cycle when $G = 3$ the next to last partial product is formed in the S-adder, and the last multiplicand is prepared for addition to the partial product during $G = 2$ time. The conditions to enable control points during $G = 3$ are the same as those used for $G > 3$ with these exceptions:

1) For the decode of $M_{62-63} = 10_2$ and STFF = 0 (multiplier = ± 2), two times the multiplicand is prepared for subtraction from the partial product during $G = 2$. This necessitates that $¢ZAQ_{36-71}/N_{36-71}$ be enabled and the control to set K_{72} must be enabled.

2) When $G = 3$, a level resets the DIC flip-flop and frees the M-register.

3) The control on the set/reset of PPS is not a function of the (± 2) multiplier.

4) STFF is unconditionally reset.

When the \$ pulse is generated at the end of $G = 3$, the same strobes occur as for $G > 3$ with the exception of \$M. \$M is not produced because the multiplier in M has been exhausted by this time. This strobe decrements the G count to 2 and the multiply continues.

GO (G = 2)

During $G = 2$ the final partial product is formed in the S-adder as a function of the former partial product in H and the multiplier controlled multiplicand in N. Because this will be the last add cycle, no control points are enabled to load a new multiplicand into N on the strobe at the end of $G = 2$. The control line ¢S_{0-69}/H_{2-71} will be enabled to transfer the final partial product to H. At the same time the length of the partial product will be increased by one bit by transferring the contents of S_{70} to H_{72}.

When the \$ pulse is generated at the end of $G = 2$, the same strobes are produced that occurred at the end of $G = 3$. Because no switches are enabled into N, this register is cleared to all zeros. The K_{72} flip-flop is unconditionally reset and the G count is decremented to one.

GO (G = 1)

During $G = 1$ the final mantissa product is contained in H_{1-72} with the binary point to the right of H_1. Control is set up to transfer H_{1-72} N_{0-71} and thereby place the binary point to the right of N_0. The control points enabled to accomplish this are ¢S_{1-71}/N_{0-70} and ¢H72/N71. At the same time controls are set up to add the two exponents. ¢D_{0-7}/ES_{0-9} and \$$ES_{0-9}/E_{0-9}$ are enabled to accomplish this.

When the $ pulse is generated at the end of G = 1, the following events occur:

1) $N transfers the product with a left shift of one from H via the S-adder to N.
2) $G decrements the G count to zero.
3) $H clears H to zero.
4) $E strobes the sum of the exponents into E.
5) SIGN is set if $S_0=1$.
6) ZERO is set if the S-adder output = 0.
7) REV is set if $(S_0\overline{S}_1 + \overline{S}_0 S_1)$.
8) GN is set if $(OFL) + (S \neq 0)(S_0 = S_1)$.
9) GF is set if $(\overline{OFL})(S = 0) + S_0 = S_1)$.
10) GO is reset.
11) $ enters the timing pulse generator delay line network to produce another $ pulse.

GN

During this cycle the mantissa in N is normalized. If an overflow occurred during the last GO cycle the mantissa is shifted one place right with opposite sign inserted and the G-counter is decremented to (-1). In all other cases the mantissa is shifted left until No and N are not alike and the G-counter is incremented by one for each place the mantissa is shifted. The control lines that become enabled to accomplish this are:

1) ¢N-1 (¢N_{0-70}/N_{1-71}) if REV set.

2) ¢N + 16 (¢N_{16-71}/N_{0-55}) if N_0 through N_{15} are all the same.

3) ¢N + 4 (¢N_{4-71}/N_{0-67}) if ¢$\overline{N + 16}$ and N_0 through N3 are allthe same.

4) ¢N + 1 (¢N_{1-71}/N_{0-70}) if ($\overline{¢N + 16}$)($\overline{¢N + 4}$) and $N_0 = N_1$.

5) LSN enabled if (¢N + 16) ($N_{15} = N_{16} \neq N_{17}$) + (¢N + 4)($N_3 = N4 \neq N5$) + (¢N + 1)($N_0 = N_1 \neq N_2$) + (REV).

At the end of 230 nanoseconds a $ pulse will cause the following events:

1) $N strobes the shifted contents of N back into N.

2) $G to increment the G-counter by 16 if ¢N + 16, by 4 if ¢ N + 4, by 1 if ¢N + 1, and decrement G by 1 if REV.

3) REV to set if (¢N + 16)($N_{15} \neq 16$) + (¢N + 4) ($N_3 \neq N_4$).

4) REV to reset if REV is set.

12.3.10.1-contd

5) GN to reset and GF to set if LSN enabled.

6) \$ enters the timing pulse generator delay line network to produce another \$ pulse.

GN will be repeated as many times as required to normalize the mantissa and will terminate on the first \$ pulse following the enabling of LSN.

GF

During this cycle the mantissa in N is transferred to AQ, and the count in G is subtracted form E, with the resultant exponent stored in E. The control lines that become enabled to accomplish this are:

1) $¢S_{0-71}/AQ_{0-71}$.

2) $¢\overline{G_{0-7}}/ES_{0-7}$ and $¢CARRY/ES_7$.

3) ¢ES/E if mantissa $\neq 0$.

4) $¢-2^7/E$ if mantissa $= 0$.

At the end of 400 nanoseconds a \$ pulse will cause the following events:

1) \$AQ will strobe the mantissa product into AQ.

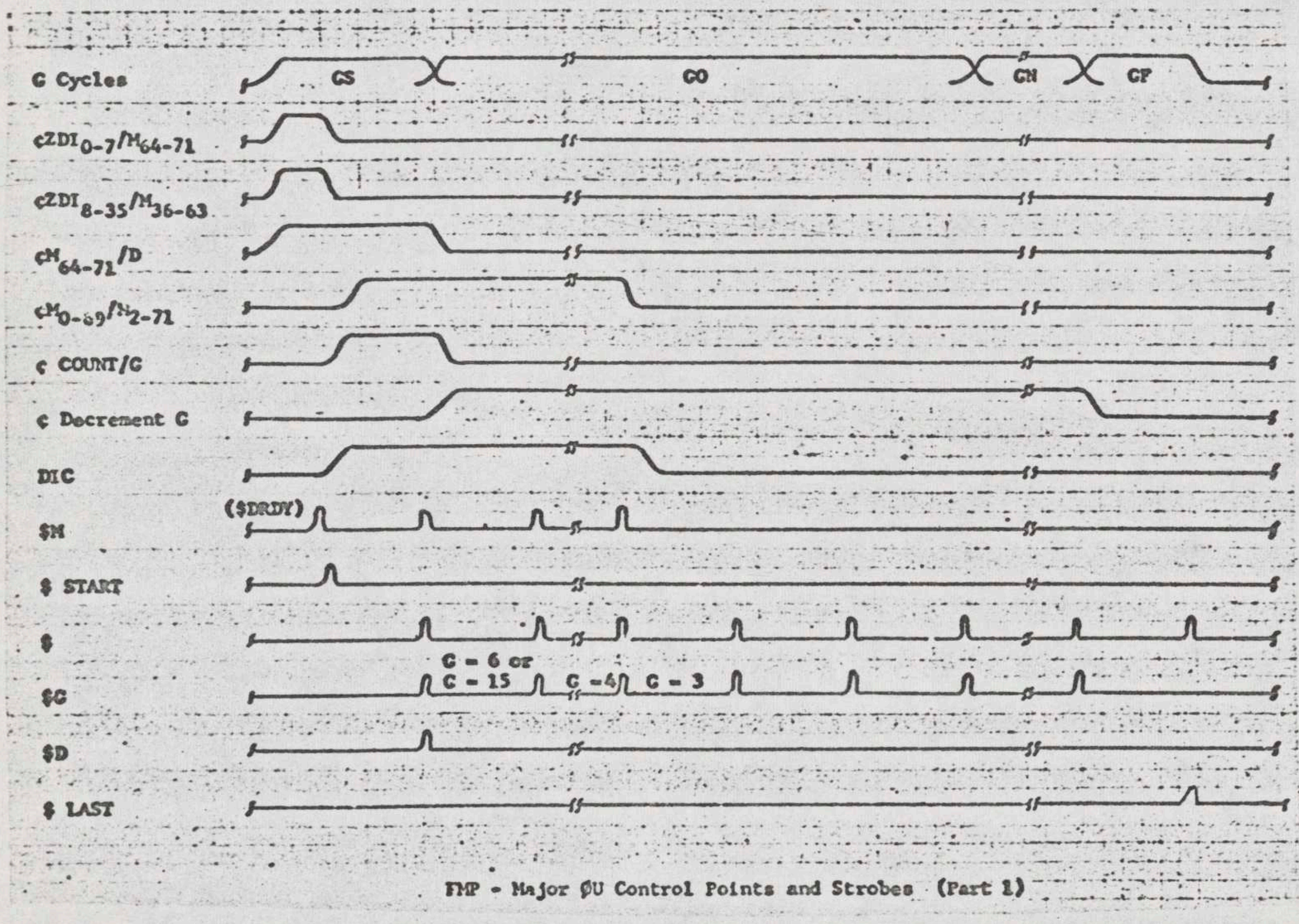
2) \$E will strobe the exponent into E.

3) EOFL will set if the subtraction of G from E caused the exponent to exceed +127.

4) EUFL will set if the subtraction of G from E caused the exponent to become smaller than -128.

5) GF will reset.

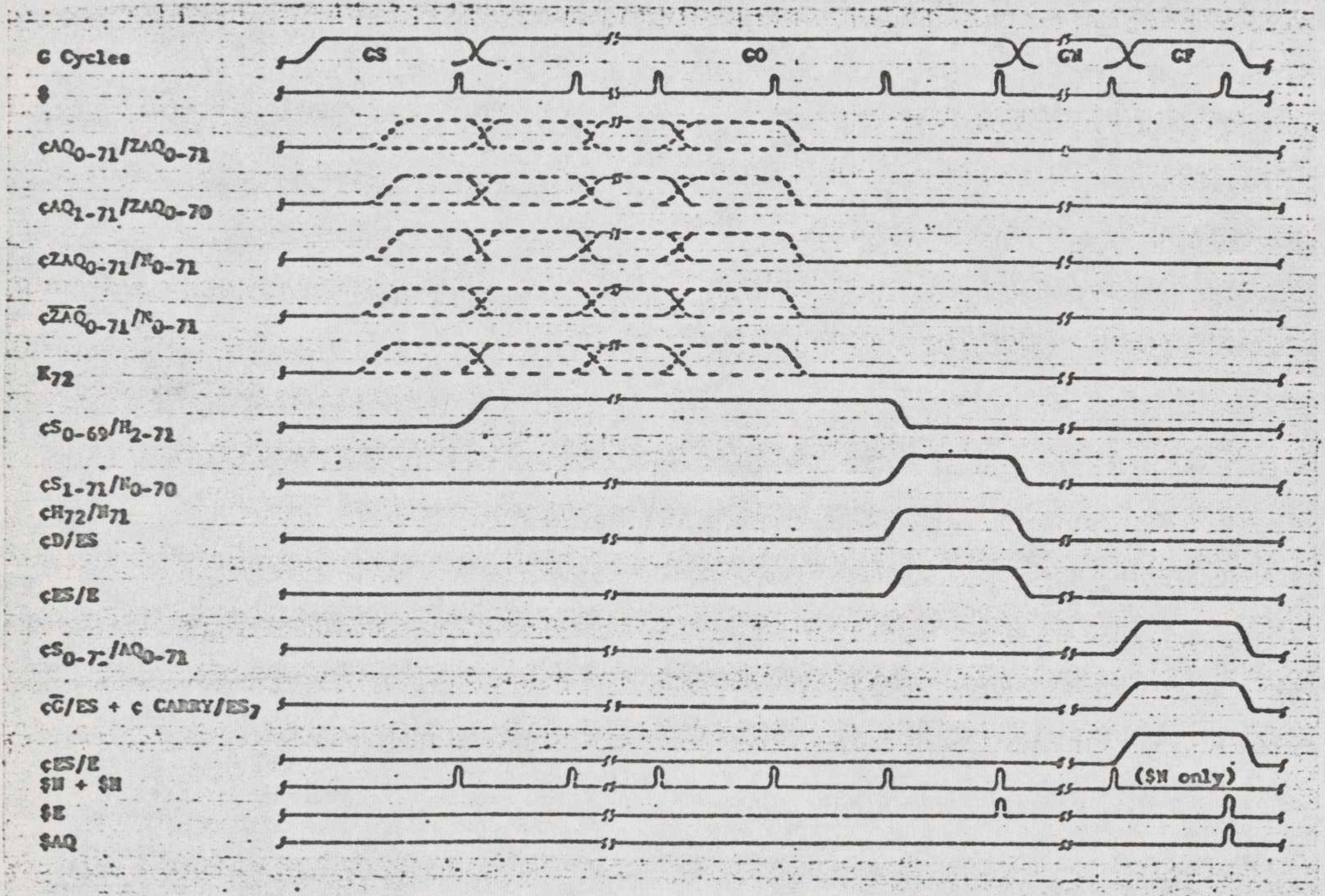
6) $ will generate a $ LAST, which is sent to the CU to indicate completion of the FMP instruction.


G Cycles
GS
GO
GN
GF
cZDI0-7/M64-71
cZDI8-35/M36-63
cM64-71/D
cM0-69/N2-71
c COUNT/G
c Decrement G
DIC
$M
($DRDY)
$ START
$
$G
G = 6 or
G = 15
G = 4
G = 3
$D
$ LAST

FMP - Major ØU Control Points and Strobes (Part 1)

Fig 12

FIG. 12.3.10.1 FMP Instruction, Timing Diagram
(Sheet 1 of 2)


C Cycles
CS
CO
CN
CF
$
cAQ0-71/ZAQ0-71
cAQ1-71/ZAQ0-70
cZAQ0-71/N0-71
cZAQ0-71/N0-71
K72
cS0-69/H2-71
cS1-71/N0-70
cH72/H71
cD/ES
cES/E
cS0-7_/AQ0-71
cG/ES + c CARRY/ES7
cES/E
$H + $H
$E
$AQ
($H only)

FMP - Major ¢U Control Points and Strobes (Part 2)

FIG. 12.3.10.1 FMP Instruction, Timing Diagram
(Sheet 2 of 2)

12.3.10.2 Double Precision

The instructions DFMP and DUFM allow double precision floating multiplication to be accomplished. For both instructions a 72-bit mantissa (multiplicand) is multiplied by a 64-bit mantissa (multiplier) to produce a 72-bit mantissa (product). At the same time the two 8-bit exponents are added to produce 8-bit product exponent. The multiplier mantissa is contained in M_{0-63} , the partial products are accumulated in H_{0-72}, and the final mantissa product is stored in AQ_{0-71}. The multiplier exponent contained in M_{64-71} is transferred to D, the multiplicand exponent is contained in E, and the exponent sum is stored in E.

The only difference between a DFMP and a DUFM instruction is in the handling of the mantissa product. During a DFMP the mantissa product is always normalized, while during a DUFM the mantissa product is normalized only if an overflow occurred.

For detailed control of these two instructions refer to the OU Specification Chart 43D140232 page 8.

The DFMP is identical to the FMP, as is the DUFM and UFM, except during GS. Only the GS cycle will be discussed here. Refer to 12.3.10.1 for all other cycles.

GS

The IM-switch is enabled by the presence of the DFMP or DUFM decode together with DPFF, and DIC reset. The control points ¢ZDI_{0-7}/M_{64-71} and ¢ZDI_{8-35}/M_{0-27} are enabled to allow the ZDI bus to be present at the proper M-register inputs.

12.3.10.2-contd

When the first $DRDY is received from the CU, DPFF reset allows $\$M_{0-71}$ to be generated and strobe the exponent and most significant 28 bits of the mantissa into M. This $DRDY will set DPFF, but will not generate a $YRY. On the setting of DPFF, the IM-switch will be unabled by $¢ZDI_{0-7}/M_{28-35}$ and $¢ZDI_{8-25}$/M36-63.

About 250 nanoseconds after the first $DRDY another $DRDY will be received. DPFF set will allow $\$M_{28-63}$ to be generated and strobe the least significant 36 bits of the mantissa into the M-register. This $DRDY resets DPFF and is turned around as $YRY and returned to the CU to signal that a new instruction may be prepared. $YRY also sets YRY, and because of the DFMP or DUFM decode sets DIC. The DIC flip-flop is used as an inhibit on the ZDI/M controls for the duration in which the multiplier is contained in M. When YRY is set the conditions exist to allow a $ START to be generated by a last of three pulse detector. $ START sets OUB, resets YRY and DDF, and transfers the operation code from the O-register to the C-register. It also enters the timing pulse generator delay line network.

During the 300 nanoseconds of GS from $ START to the first $, control is established to load the G-counter with a count of 33 on the first $. $¢M_{64-71}/D$ becomes enabled to transfer the exponent to D.

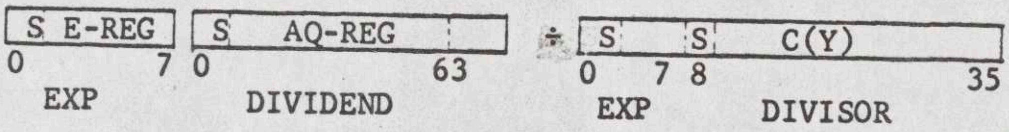
Everything from this point on is the same as for FMP instruction.

12.3.11 FLOATING POINT DIVIDE GROUP

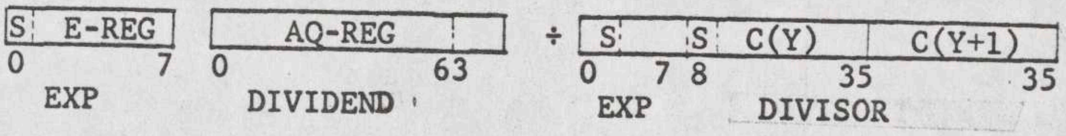
This group of four instructions comprise the floating point divide capabilities of this machine. The major control points and data paths can be seen in figure 12.3.11.

The instructions FDV, FDI, DFDV, and DFDI give the capabilities of dividing by 28-bit (single precision) and 64-bit (double precision) mantissas, including sign. In each case there are eight bits of exponent including sign.

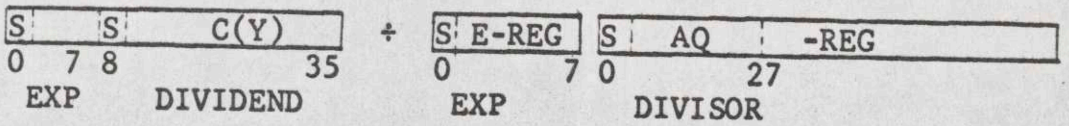
FDV


S E-REG
0
7
EXP
S
AQ-REG
0
63
DIVIDEND
÷
S
S
C(Y)
0
7
8
35
EXP
DIVISOR

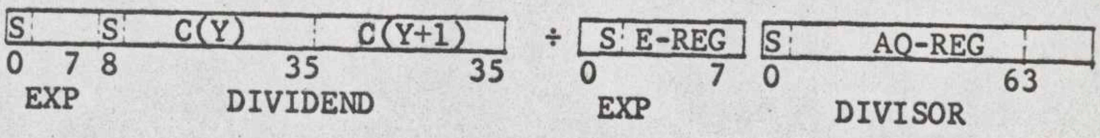
DFDV


S
E-REG
0
7
EXP
AQ-REG
0
63
DIVIDEND
÷
S
S
C(Y)
C(Y+1)
0
7
8
35
35
EXP
DIVISOR

FDI


S
S
C(Y)
0
7
8
35
EXP
DIVIDEND
÷
S
E-REG
0
7
EXP
S
AQ
-REG
0
27
DIVISOR

DFDI


S
S
C(Y)
C(Y+1)
0
7
8
35
35
EXP
DIVIDEND
÷
S E-REG
0
7
EXP
S
AQ-REG
0
63
DIVISOR

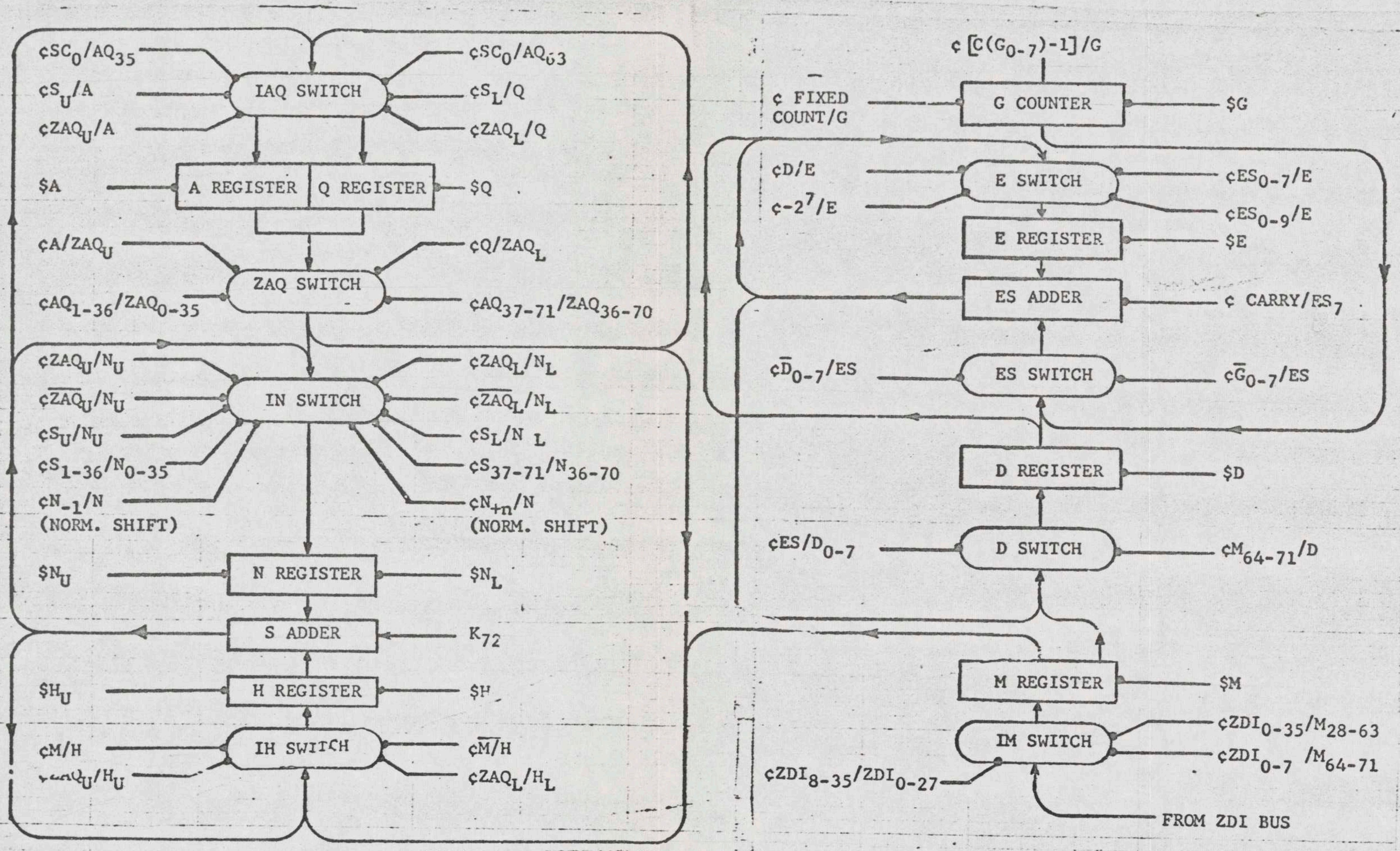

¢SC_0/AQ_{35}
¢S_U/A
¢ZAQ_U/A
IAQ SWITCH
¢SC_0/AQ_{63}
¢S_L/Q
¢ZAQ_L/Q
$A
A REGISTER
Q REGISTER
$Q
¢A/ZAQ_U
ZAQ SWITCH
¢Q/ZAQ_L
¢AQ_{1-36}/ZAQ_{0-35}
¢AQ_{37-71}/ZAQ_{36-70}
¢ZAQ_U/N_U
¢$\overline{ZAQ}_U/N_U$
¢S_U/N_U
¢S_{1-36}/N_{0-35}
¢N_{-1}/N
(NORM. SHIFT)
IN SWITCH
¢ZAQ_L/N_L
¢$\overline{ZAQ}_L/N_L$
¢$S_L/N\ L$
¢S_{37-71}/N_{36-70}
¢N_{+n}/N
(NORM. SHIFT)
$\$N_U$
N REGISTER
$\$N_L$
S ADDER
K_{72}
$\$H_U$
H REGISTER
$H
¢M/H
IH SWITCH
¢$\overline{M}/H$
¢ZAQ_U/H_U
¢ZAQ_L/H_L
¢ $[C(G_{0-7})-1]/G$
¢ FIXED COUNT/G
G COUNTER
$G
¢D/E
¢$-2^7/E$
E SWITCH
¢ES_{0-7}/E
¢ES_{0-9}/E
E REGISTER
$E
ES ADDER
¢ CARRY/ES_7
¢$\overline{D}_{0-7}/ES$
ES SWITCH
¢$\overline{G}_{0-7}/ES$
D REGISTER
$D
¢ES/D_{0-7}
D SWITCH
¢M_{64-71}/D
M REGISTER
$M
IM SWITCH
¢ZDI_{0-35}/M_{28-63}
¢ZDI_{0-7}/M_{64-71}
¢ZDI_{8-35}/ZDI_{0-27}
FROM ZDI BUS

FIG. 12.3.11 FLOATING DIVIDE GROUP REGISTERS AND DATA FLOW

12.3.11.1 Single Precision

FDV

During the instruction FDV, a 63 bit fractional dividend mantissa plus sign in the AQ-register and exponent in the E-register are divided by a 27 bit fractional divisor mantissa plus sign and exponent from a memory location Y, to produce a 35 bit fractional quotient mantissa plus sign in the A-register and an exponent in the E-register. The Q-register is cleared to zero.

The load path for the divisor into the M-register places the mantissa and sign from ZDI_{8-35} into M_{0-27} and the exponent from ZDI_{0-7} into M_{64-71}. The generation of \$ SET GS and \$ START are as developed in paragraph 12.2.

GS

During GS, the ZAQ and IN switches are enabled to convert the C(AQ) to a positive dividend. The inputs to the N-register are strobed with the signal $\$N_{0-35}$ and $\$N_{36-71}$. The control points $\text{¢}AQ_{0-35}/ZAQ_{0-35}$ and $\text{¢}AQ_{36-71}/ZAQ_{36-71}$ are activated, and if the sign of the dividend mantissa in AQ is already positive ($A_0=0$), the control points activated are $\text{¢}ZAQ_{0-35}/N_{0-35}$ and $\text{¢}ZAQ_{36-71}/N_{36-71}$ and ¢ reset K_{72}. If the sign of the dividend mantissa is negative ($A_0=1$), the control points $\overline{\text{¢}ZAQ_{0-35}}/N_{0-35}$, $\overline{\text{¢}ZAQ_{36-71}}/N_{36-71}$ and set K_{72} are activated to complement AQ and thus convert the dividend to positive. At the same time $\text{¢}M_{64-71}/D_{0-7}$ is enabled, to place the divisor exponent in the D-register.

12.3.11.1 - contd

When the \$ START pulse propogates the medium delay elements, the first operate strobe \$ occurs. This allows the following events:

1) The GS cycle to reset and the GD1 cycle to set.

2) The quotient sign flip-flop (QSF) and Zero indicator to reset.

3) \$N strobes either ZAQ or $\overline{\text{ZAQ}}$ into the N-register.

4) \$ H is generated, but since no IH switch is enabled, all zeros are loaded into H.

5) \$ D strobes the dividend exponent into the D-register.

6) The K_{72} adder control flip-flop is set if it is necessary to complement the C(AQ) to produce a positive dividend.

7) The \$ strobe re-enters the Timing Pulse generator delay line network to produce another \$ strobe in about 400 nanoseconds.

GD1

During GD1 the positive dividend will become available at the adder output. The IAQ switch is enabled and the AQ-register is strobed with the positive dividend.

The IAQ switch is enabled by ¢S_{0-35}/AQ_{0-35} and S_{36-71}/AQ_{36-71}. The IH switch is enabled by ¢M_{0-35}/H_{0-35} and ¢M_{36-63}/H_{36-63} if $M_0=1$, or by ¢$\overline{M}_{0-35}/H_{0-35}$ and ¢$\overline{M}_{36-71}/H_{36-71}$ if $M_0=0$. When the \$ strobe occurs at the end of GD1, the following events take place:

12.3.11.1 - contd

1) The GD1 cycle is reset and GD2 cycle is set.

2) $\$N_{0-71}$ is generated and loads all zeros into N since no IN switch is enabled.

3) $\$H$ is generated transferring the divisor into H.

If the original divisor was positive the one's complement of M is strobed into H at this time and K_{72} is set to create an effective negative divisor. This is retained until the G=2 cycle.

4) ¢S_{0-35}/AQ_{0-35}, ¢S_{36-71}/AQ_{36-71} and $\$AQ_{0-35}$, $\$AQ_{36-71}$ is generated to load the positive dividend into the AQ-register.

5) The Quotient Sign flip-flop (QSF) is set if the signs of the original dividend and divisor were different as indicated by $M_0 \neq A_0$.

6) The $\$$ strobe re-enters the Timing Pulse Generator delay line network to produce another $\$$ in about 400 nanoseconds.

GD2

During GD2 the negative divisor will be produced in the S adder and become available on the S bus. Switches are enabled during this time to transfer the positive dividend from AQ to N, The G counter is set to a count of 38 if the divisor is not equal to zero, and the Compare Magnitude flip-flop is set for comparison of the dividend and divisor during G0. The Zero indicator is set if the divisor is zero, and a Divide Check Fault is generated.

12.3.11.1 - contd

The ZAQ and IN switches are enabled by ¢AQ_{0-35}/ZAQ_{0-35}, ¢AQ_{36-71}/ZAQ_{36-71}, ¢ZAQ_{0-35}/N_{0-35} and ¢ZAQ_{36-71}/N_{36-71}.

When the $ strobe occurs at the end of GD2, the following events take place:

1) The GD2 cycle is reset and the G0 cycle is set.

2) $N is generated which strobes the divisor into the N-register.

3) $\$G_{0-7}$ is generated which strobes the count of 38 into the G-counter if the divisor is not zero ($S \neq 0$).

4) The Compare Magnitude flip-flop (CMFF) is set.

5) The Zero Indicator is set, and the Divide Check Fault strobe ($DVCK) is sent to the BF if the divisor is zero (S=0). The Abort flip-flop is set in the OU and the OU is brought to an initialized state.

6) If a Divide Check did not occur because of a zero divisor, the $ strobe re-enters the Timing Pulse Generator to produce another $ strobe in about 400 nanoseconds.

G0 (CMFF)

The magnitude of the divisor and dividend mantissas are compared in the S-adder at this time to determine whether the actual divide process is ready to begin. One of two conditions will exist and operation of the G0 (CMFF) cycle will be affected as follows:

12.3.11.1 - contd

1) If the |dividend mantissa| ≥ |divisor mantissa|, the divide operation is not ready to begin.

In this case, the dividend mantissa is shifted right one bit position and the dividend exponent is increased by one. The CMFF will remain set and the G-counter will not be decremented. The control points enabled for this are ¢N_{0}/N_{0}, ¢N_{0-34}/N_{1-35}, ¢N_{35-70}/N_{36-71} to shift the dividend mantissa right, and ¢ $CARRY/ES_{7}$, ¢ES_{0-9}/E_{0-9} to increment the dividend exponent.

When $ strobe occurs at the end of this G0 (CMFF) cycle the following events occur:

1) $N is generated to complete the right shift of the dividend.

2) $\$E_{0-9}$ is generated to increment the dividend exponent register.

2) If the |dividend mantissa| < |divisor mantissa|, the the divide operation is ready to begin.

In this case, the dividend mantissa will be shifted left one bit position, the AQ-registers will be cleared, and the G-counter will be decremented by one. The control points enabled for this are ¢N_{1-35}/N_{0-34}, ¢N_{36}/N_{35}, ¢N_{37-71}/N_{36-70} to shift the dividend mantissa left, and ¢FCD to decrement the G-counter.

When the $ strobe occurs at the end of this G0 (CMFF) cycle, the following events occur:

12.3.11.1 - contd

1) $\$N$ is generated which enables the left shift of the dividend mantissa.
2) $\$AQ_{0-71}$ clears the AQ-register.
3) $\$G_{0-7}$ enables decrementation of the G-counter.
4) $\$$ and $\overline{SC_0}$ will reset the CMFF flip-flop.

G0 (G>3)($\overline{CMFF}$)

During the next 34 G0 cycles or until the G-counter equals 3, all but the last of the divide subcycles are completed to produce the quotient in the A-register. For each of these 34 G0 cycles the magnitude of the divisor mantissa in the H-register is subtracted from the magnitude of the dividend mantissa in the N-register, the subtraction taking place in the S-adder. If there is a carry from the most significant bit of the S-adder ($SC_0=1$) indicating that the $|$current dividend mantissa$| \geq |$divisor mantissa$|$, or in other words, that the current dividend is divisible by the divisor, then the adder result (remainder) is shifted left one bit and transferred to the H-register to become the new dividend for the divide operation to be performed in the next G0 cycle. The control points enabled for this are $¢S_{1-36}/N_{0-35}$ and $¢S_{37-71}/N_{36-70}$. If there is no carry from bit zero of the adder (the current, dividend mantissa < divisor mantissa or in other words, the dividend mantissa is not divisible by the divisor mantissa), the contents of the N-register (current dividend mantissa) are shifted left one bit position to form a new dividend for the divide operation to be performed in the next G0 cycle. The control points enabled for this are $¢N_{1-35}/N_{0-34}$, $¢N_{36}/N_{35}$, and $¢N_{37-71}/N_{36-70}$.

12.3.11.1 - contd

At the end of each of the 34 G0 (G>3) cycles, a new partial quotient is formed. This is accomplished by shifting the contents of the AQ-register left one bit position and transferring the contents of the S-adder carry output (SC$_0$) to bit 35 of the A-register. Thus the quotient is generated, one bit per G0 (G>3) cycle with the most significant bit being generated first. The control points enabled for this are ¢AQ$_{1-36}$/ZAQ$_{0-35}$, ¢AQ$_{37-71}$/ZAQ$_{36-70}$, ¢ZAQ$_{0-35}$/AQ$_{0-35}$, and ¢SC$_0$/AQ$_{35}$. ¢FCD also is present to enable decrementation of the G-counter by one.

When the $\$$ strobe occurs at the end of each G0(G>3) cycle the following events occur:

1) $\$$N enables the loading of a new dividend into the N-register, the generation of the new dividend being dependent on the state of the carry from the adder bit zero.

2) $\$$AQ$_{0-35}$ enables formation of a new partial quotient thru shifting of the old partial quotient left one bit position and loading a new quotient bit into AQ$_{35}$.

3) $\$$G$_{0-7}$ decrements the G-counter by one.

4) The $\$$ strobe re-enters the Timing Pulse generator to produce a subsequent $\$$ 320 nanoseconds later.

12.3.11.1 - contd

G0 (G=3)

At the time the G-counter is decremented to G=3, the last subtraction of the divisor mantissa from the dividend mantissa has been performed. In this cycle, the result of the last subtraction is used to form the last partial quotient. The last partial quotient is formed by shifting the contents of the AQ-register left one bit position and loading the content of the S-adder carry output (SC_0) into bit 35 of the A-register. The K_{72} flip-flop is reset, the H-register is cleared, and the G-counter is decremented by one. The control points enabled for this are ¢AQ_{1-36}/ZAQ_{0-35}, ¢AQ_{37-71}/ZAQ_{36-70}, ¢ZAQ_{0-35}/AQ_{0-35}, ¢ZAQ_{36-71}/AQ_{36-71}, ¢SC_0/AQ_{35}, and ¢FCD. It should be noted that the control point ¢SC_0/AQ_{35} overrides the ¢ZAQ/AQ for bit 35.

When the $ strobe occurs at the end of G0 (G=3), the following events occur:

1) $$AQ_{0-35}$ enables shifting and loading the last partial quotient in the A-register.

2) $$H_{0-71}$ clears the H-register since no input switches are activated.

3) $$G_{0-7}$ enables G-counter decrementation.

4) $ resets K_{72}.

5) The $ strobe propogates the medium delay.logic and appears after a delay of 320 nanoseconds.

12.3.11.1 - contd

GØ (G=2)

During the GØ (G=2) cycle the Quotient Sign flip-flop (QSF) determines whether the final quotient must be sign-corrected. If the quotient must be corrected to negative (QSF=1), the complement of the AQ-register is transferred into the N-register and the K_{72} flip-flop is set. The control points enabled for this are ¢AQ_{0-35}/ZAQ_{0-35}, ¢AQ_{36-71}/ZAQ_{36-71}, ¢ $\overline{ZAQ_{0-35}}/N_{0-35}$, ¢$\overline{ZAQ}_{36-71}/N_{36-71}$. If the quotient does not need to be sign corrected (QSF=0), the contents of the AQ-register are transferred to the N-register. The control points enabled for this are ¢AQ_{0-35}/ZAQ_{0-35}, ¢AQ_{36-71}/ZAQ_{36-71}, ¢ZAQ_{0-35}/N_{0-35}, and ¢ZAQ_{36-71}/N_{36-71}.

When the $ strobe occurs at the end of the GØ (G=2) cycle the following events occur:

1) $N enables the quotient (complemented or not) to be loaded into the N-register.

2) $$G_{0-7}$ enables decrementing of the G-counter.

3) The $ strobe re-enters the Timing Pulse Generator to produce another $ after a delay of 400 nanoseconds.

12.3.11.1 - contd

G0 (G=1)

At the time the G-counter equals one the divisor exponent is subtracted from the dividend exponent, and the difference is placed in the E-register. The control points enabled for this are ¢D_{0-7}/ES_{0-9} and ¢ CARRY/ES_7. Should the Zero indicator be set, indicating a quotient mantissa of zero, the E-register bit zero is forced to a one state ($E_0=1$) replacing the exponent difference. This is done to indicate a division result of floating point zero, defined as 0×2^{-128}. This is accomplished by enabling ¢-2^7/E_{0-7}. The quotient is returned to the AQ-register via ¢S_{0-35}/A_{0-35} and the Sign and Zero indicators are set accordingly.

If the quotient is non-zero ($S\neq 0$) and $S_0\neq S_1$, it is sent to the AQ-register via ¢S_{0-35}/AQ_{0-35}. Should $S_0=S_1$, indicating an unnormalized quotient, it is retained for normalization and not sent to the A-register at this time.

When the $ strobe occurs at the end of G0 (G=1) cycle the following takes place:

1) The Sign indicator is set if bit zero of the S-adder is a one ($S_0=1$), otherwise it is reset.

2) The E-register is strobed with the output of the ES-adder.

3) K_{72} is reset.

4) If the quotient is zero ($S=0$) or if the quotient is nonzero and bits zero and one of the S-adder are different ($S\neq 0$)($S_0\neq S_1$) the final quotient is strobed into the A-register by $\$AQ_{0-35}$.

12.3.11.1 - contd

5) The Zero Indicator is set if S=0.

6) The GØ cycle flip-flop is reset and the GN cycle flip-flop is set if $(S\neq0)(S_0=S_1)$, or the GF cycle flip-flop is set if S=0 or if $S_0\neq S_1$.

7) The $ strobe propogates the slow delay logic after 400 nanoseconds during GF, or it exits the fast delay logic after 230 nanoseconds during GN.

GN

The normalization of the mantissa during a FDU is identical to the action described during GN of a FNØ instruction, see section 3.3.12.1.

GF

If the GF cycle was preceeded by a GN, the GF of the FDU is identical to the GF of the instruction FNØ, see section 12.3.12.1.

Should the resulting quotient have been found to be normalized during (GØ)(G=1) the only activity as a result of GF is the generation of a $ LAST. This pulse signals the control frame (CU) that the operations unit (ØU) has completed an instruction.

FDI

The instruction Floating Divide Inverted operates as the name implies. The contents of the A-register, bits zero to twenty eight, being the divisor mantissa and the contents of some location Y, bits eight to thirty-five, becoming the dividend mantissa. This is accomplished by switching the contents of the A-register with the H-register during the GD2 cycle.

12.3.11.1 - contd

Exponent manipulation is also reversed by subtracting the C(E$_{0-7}$) from some location Y bits 0-7.

The loading of the dividend is accomplished through the control points ¢ZDI$_{8-35}$/M$_{0-27}$ and ¢ZDI$_{0-7}$/M$_{64-71}$.

GS

The action during this timing cycle is the same as during the FDV instruction, except that the C(AQ) are converted to a negative mantissa.

GD1

As in GS the events during the GD1 cycle are the same as in the FDV instruction, except that the C(M) is converted to a positive mantissa.

GD2

During the GD2 divide cycle the contents of the AQ and H-registers and the E and D-registers are reversed. This is accomplished by enabling the following control points: ¢S$_{0-35}$/N$_{0-35}$, ¢S$_{36-71}$/N$_{36-71}$, ¢AQ$_{0-35}$/ZAQ$_{0-35}$, ¢AQ$_{36-71}$/ZAQ$_{36-71}$, ¢ZAQ$_{0-35}$/H$_{0-35}$, ¢ZAQ$_{36-71}$/H$_{36-71}$, ¢ES$_{0-7}$/D$_{0-7}$, ¢D$_{0-7}$/E$_{0-9}$.

When the \$ occurs at the end of the GD2 cycle the following events take place:

1) The dividend mantissa is strobed into the N-register.

12.3.11.1 - contd

2) The divisor mantissa is strobed into the H-register.

3) The dividend exponent is strobed into the E-register.

4) The divisor exponent is strobed into the D-register.

5) K_{72} is reset.

6) The G0 cycle flip-flop is set, and the CMFF flip-flop is set.

The remaining portion of the instruction is the same as the sequence during FDV.

12.3.11.2 Double Precision

DFDV

The Double Precision Floating Divide instruction (DFDV) is basically the same as the Floating Divide Instruction (FDV) with the following exceptions.

1) The divisor mantissa consists of $C(Y)_{8-35}$ and $C(Y+1)_{0-35}$.

2) The G-counter is set to a count of 66 during the GD2 cycle.

3) The carry out of S_0 will be inserted into AQ_{63}.

4) The quotient mantissa will be contained in $C(AQ)_{0-63}$ at the end of GF.

For a detailed explanation see section 12.3.11.1.

DFDI

The Double Precision Floating Divide Inverted (DFDI) is identical to the Floating Divide Inverted (FDI) instruction with the following exceptions:

1) The dividend mantissa consists of the $C(Y)_{8-35}$ and $C(Y+1)_{0-35}$.

2) The divisor mantissa is contained in the $C(AQ)_{0-63}$.

3) The G-counter is set to a count of 66 during the GD2 cycle.

12.3.11.2 - contd

4) The carry out of S_0 will be inserted into AQ_{63}.

5) The quotient mantissa will be contained in the $C(AQ)_{0-63}$ at the end of GF.

For a detailed explanation see section 12.3.11.1.

12.3.12 FLOATING POINT SPECIAL INSTRUCTIONS

This group of two instructions completes the floating point repertoire. They are FNO and FNEG.

12.3.12.1 FNO (Floating Normalize)

Generally, floating point numbers are normalized by shifting the mantissa left and correspondingly decreasing the exponent until no leading zeros are present in the value portion of the mantissa for positive numbers or until no leading ones are present in the value normalize mantissa for negative numbers. Zeros fill in the bit positions vacated on the right in either case.

$$(AQ)_{0-71} \xleftarrow[X]{L} (AQ)_{0-71}$$

$$E_{0-7} - X \Longrightarrow (E_{0-7})$$

L denotes left shift
X denotes the number of shifts required to first satisfy the condition $\bar{A}_0 A_1 + A_0 \bar{A}_1$

In the special case of OFLI being set when FNO is entered, the FNO instruction is executed by shifting the mantissa to the right one bit position, replacing the vacated bit position (sign bit) with the inverse of its previous contents, and incrementing the exponent by one as expressed below:

$$(AQ)_{0-70} \Longrightarrow (AQ)_{1-71}, \ (A_0) \Longrightarrow (A_0)$$

and $E_{0-7} + 1 \Longrightarrow E_{0-7}$

12.3.12.1 - contd

GS

During the GS cycle the H-register is cleared. The contents of the AQ-register are gated through the ZAQ bus and into the N-register. The Sign and Zero flip-flops are both reset and at the end, the GO flip-flop is set.

GO

If the overflow indicator is set as a result of some previous operation, the REV flip-flop will be set.

The operand is examined in the S-adder (outputs of the N- and H-registers are constantly fed to the S-adder and since the H-register was cleared during GS, the S-adder contains only the operand during GO), and if the contents of the S-adder are not zero and $S_0 = S_1$, or if the overflow indicator is set from some previous operation, the GN flip-flop will be set which will start the normalize cycle. If $S_0 = 1$, the Sign flip-flop will be set. If the S-adder contains all zeros, the Zero flip-flop will be set.

The G-counter will be cleared by strobing while no input controls are active.

The GN cycle is now entered if the GN flip-flop is set.

GN

During the GN cycle, the contents of the N-register may be left undisturbed (first GN cycle only), shifted right one, or shifted left one, four or sixteen bit positions. This shifting is accomplished in the N-register with its associated shifting logic. The conditions which control shifting, together with resulting action are tabulated below:

12.3.12.1 - contd

CONDITIONS	ACTION
1) REV FF Set	$N_{0-70} = N_{1-71}$, $N_0 = N_0$ (Shift right one)
2) $[GN][\overline{REV}][\overline{ZERO}][(N_0 \cdots N_{15}) + (\overline{N}_0 \cdots \overline{N}_{15})]$ (Bits 0-15 of the N-register are alike, the REV flip-flop is not set and the ZERO flip-flop is not set.)	$N_{16-71} = N_{0-55}$, $0 = N_{56-71}$ (Shift left sixteen)
3) $[GN][\overline{REV}][\overline{ZERO}][(N_0 \cdots N_3) + (\overline{N}_0 \cdots \overline{N}_3)]\overline{[(N_0 \cdots N_{15}) + (\overline{N}_0 \cdots \overline{N}_{15})]}$ (Bits 0-3 of the N-register are alike, the REV flip-flop is not set, the ZERO flip-flop is not set, and bits 0-15 of the N-register are not alike.)	$N_{4-71} = N_{0-67}$, $0 = N_{68-71}$ (Shift left four)
4) $[GN][\overline{REV}][\overline{ZERO}][N_0 N_1 + \overline{N}_0 \overline{N}_1]\overline{[(N_0 \cdots N_3)+(\overline{N}_0 \cdots \overline{N}_3)]}$ (Bits 0 and 1 of the N-register are alike, the REV flip-flop is not set, ZERO flip-flop is not set, and bits 0-3 of the N-register are not alike.)	$N_{1-71} = N_{0-70}$, $0 = N_{71}$ (Shift left one)

12.3.12.1 - contd

If a shift of four or sixteen is required, the possibility of an overshift of one bit exists.

For example, if bits 0-3 are all zeros and bit 4 is a one, a left shift of four will be made. In this case, the sign bit will be changed from plus to minus, which case is included in the definition of overshift.

If the shifting action called for will result in an overshift, the REV flip-flop gets set in the current GN cycle so that a right shift of one will be executed in the next GN cycle to correct for the overshift.

The fact that an overshift will be made in the current GN cycle is detected by implementation of the following logic expression:

$$[GN][\overline{REV}][\overline{ZERO}][(N_0 \cdots N_{15})+(\overline{N}_0 \cdots \overline{N}_{15})][N_0\overline{N_{16}} + \overline{N_0}N_{16}] +$$
$$[GN][\overline{REV}][\overline{ZERO}][(N_0 \ldots N_3)+(\overline{N}_0 \ldots \overline{N}_3)][N_0\overline{N}_4 + \overline{N}_0 N_4]$$

The information needed for properly decrementing the exponent register is retained in the G-counter. The number of shifts made in each GN cycle are added to the contents of the G-counter by means of control logic and appropriate control signals. In the case where a right shift is required due to a previous overflow or an overshift in a previous GN cycle, the signal "Full Counter Decrement" is applied and causes the G-counter to be decremented by one.

Generation of the signal LSN (last normalize shift) determines whether the normalization is complete with the current GN cycle or whether further GN cycle(s) are required. The following logic is implemented to generate LSN.

12.3.12.1 - contd

$$
\begin{aligned}
\mathrm{LSN} = [&(\mathrm{N\ DET\ 1})(N_1\overline{N}_2 + \overline{N}_1 N_2)+(\mathrm{N\ DET\ 4_B})(\overline{N}_4 N_5) + \\
&(\mathrm{N\ DET\ 4_B})(N_4\overline{N}_5)+(\mathrm{N\ DET\ 16_B})(\overline{N}_{16} N_{17}) + \\
&(\mathrm{N\ DET\ 16_A})(N_{16}\overline{N}_{17})+(N_0\overline{N}_1 + \overline{N}_0 N_1) + \mathrm{REV}]\ \mathrm{GN}
\end{aligned}
$$

This specifies that if the bits which will replace bits N_0 and N_1 in this shift cycle are not equal, normalization of the mantissa will be complete with this shift cycle.

A GF cycle follows the last GN cycle.

GF

At the end of the last GN cycle, the normalized mantissa is contained in the S adder since the contents of the N-register and the H-register are constantly being summed and the H-register was cleared during the GS cycle.

The normalized mantissa is gated and strobed into the AQ-register from the S-adder.

The contents of the G-counter are gated to the ES-(Exponent) adder together with a carry bit to the ES_7 for addition to the contents of the ES-register This addition of the two's complement of the contents of the G-counter effects a subtraction of it from the contents of the ES-register. The resulting contents of ES_{0-7} are gated and strobed into the ES-register if the ZERO indicator is not set. If the ZERO indicator is set, a one is forced into bit 0 of the ES-register. This yields an exponent equal to -2^7 or -128 which, with a mantissa of 0.00...0, has been defined as the machine representation of the number zero.

If an exponent underflow or overflow exists, the appropriate flip-flop will be set.

12.3.12.2 FNEG (Floating Negate)

In general, the Floating Point Negate instruction is performed by forming the two's complement of the mantissa contained in the AQ-register and then normalizing the resultant.

GS

During the GS cycle the H-register is cleared by strobing with no input control activated.

The contents of the AQ-register are gated to the ZAQ bus and $\overline{ZAQ}$ is gated and strobed into the N-register. This puts the one's complement of the mantissa into the N-register.

The Sign and Zero flip-flops are both reset and K_{72} flip-flop is set.

The outputs of the N- and H-registers are constantly summed in the S-adder and K_{72} is also added into the least significant bit position. Since the H-register is cleared to zero, adding K_{72} to the one's complement of the operand results in the two's complement in the S-adder.

GO

During the GO cycle the contents of the S-adder are gated and strobed into the N-register.

The REV flip-flop is set if the overflow condition is present. That is, if $\overline{C_0}C_1 + C_0\overline{C_1}$ is true.

The GN flip-flop will be set if the overflow condition is present or if the contents of the S-adder are not zero and S_0 and S_1 are alike (indicating that the number is not normalized).

12.3.12.2 - contd

The Sign indicator will be set according to the following logic:

$$\text{Set SIGN} = S_0 \cdot \overline{\text{OVERFLOW CONDITION}} + \overline{S}_0 \cdot \text{OVERFLOW CONDITION}$$

The Zero flip-flop will be set if the contents of the S-adder are zero.

K_{72} flip-flop will be reset.

GN-GF

If the GN flip-flop is set, the next cycle will be a GN cycle, otherwise the next cycle will be a GF cycle. In either case, the action from this point is identical to that described for the same timing cycles of the FNO instruction except that if the overflow indicator is set, it does not get reset in the GF cycle as it does for a FNO instruction.

12.4 G-COUNTER DESCRIPTION

The G-counter provides:

1) Sequence control for all instructions requiring multiple operation cycles. For example, multiply, divide, and GTB instructions.

2) Control for executing the specified number of shifts and the bits per shift cycle sequence during shift instructions.

3) Control for executing the required number of mantissa alignment shifts and the bits per align cycle sequence for floating instructions.

4) Register capability for accumulating the number of bit positions a mantissa is shifted during a normalization cycle.

The G-counter is an 8-bit counter/register which can assume the following logical configurations dependent upon the function it is performing:

1) Eight-bit decremental counter when providing sequence control for multiply, divide, and GTB instructions. The number of operation cycles required by each instruction is inserted in the counter and decremented by one after each operation cycle.

   Other special functions require this full counter decrement-by-one capability and are discussed in paragraph 12.4.13.

12.4 - contd

2) Three individual counters each having decremental capability. Shifts and/or aligns are performed by shifting either 16-, 4-, or 1-bit(s) per cycle. The number of shifts of 16 are defined by bits G_{1-3}, shifts of 4 by G_{4-5}, and shifts of 1 by G_{6-7}. Each of these counters is individually decremented in a 16, 4, 1 order.

3) Three individual counters each having incremental capability, as a parallel to the execution of shifts or aligns. Normalization shifts are accrued by the G-counter in a 16, 4, 1 order. The number of shifts of 16 are accrued in G_{1-3}, shifts of 4 in G_{4-5}, and shifts of 1 in G_{6-7}. The total number of normalization shifts is sent to the exponent section to perform exponent correction.

12.4.1 G-COUNTER CONTROL DURING MULTIPLY, DIVIDE AND GTB INSTRUCTIONS

12.4.1.1 Operation Cycles Required

The following table lists the number of operation cycles required for each operation and the timing cycle during which this number is inserted in the G-counter. This number will be decremented by one at the end of each GO cycle.

FIXED COUNT INSERTION

Inst.	Set G =	Timing Cycle	$G_1G_2G_3G_4G_5G_6G_7$
FMP	15	GS	0 0 0 1 1 1 1
UFM	15	GS	0 0 0 1 1 1 1
DFMP	33	GS	0 1 0 0 0 0 1
DUFM	33	GS	0 1 0 0 0 0 1
FMP_{DU}	6	GS	0 0 0 0 1 1 0
UFM_{DU}	6	GS	0 0 0 0 1 1 0
DFMP_{DU}	6	GS	0 0 0 0 1 1 0
DUFM_{DU}	6	GS	0 0 0 0 1 1 0
MPF	19	GS	0 0 1 0 0 1 1
MPY	19	GS	0 0 1 0 0 1 1
MPF_{DU}	10	GS	0 0 0 1 0 1 0
MPF_{DL}	19	GS	0 0 1 0 0 1 1
MPY_{DU}	10	GS	0 0 0 1 0 1 0
MPY_{DL}	19	GS	0 0 1 0 0 1 1
DVF	38	GD2	0 1 0 0 1 1 0
DIV	38	GD2	0 1 0 0 1 1 0
FDV	38	GD2	0 1 0 0 1 1 0
DFDV	66	GD2	1 0 0 0 0 1 0
FDI	38	GD2	0 1 0 0 1 1 0
DFDI	66	GD2	1 0 0 0 0 1 0
FDV_{DU}	38	GD2	0 1 0 0 1 1 0
DFDV_{DU}	66	GD2	1 0 0 0 0 1 0
FDI_{DU}	38	GD2	0 1 0 0 1 1 0
DFDI_{DU}	66	GD2	1 0 0 0 0 1 0
BCD	6	GD2	0 0 0 0 1 1 0
GTB	36	GS	0 1 0 0 1 0 0

DU = Direct Upper Modifier

DL = Direct Lower Modifier

12.4.1.2 Fixed Count Insertion Control Equations

Decode: ① = V_4V_7 = DFDV + DFDI

② = $V_4\overline{V}_7$ = FDV + FDI

③ = $\overline{V}_4V_7$ = DVF + DIV

④ = $\overline{V}_4\overline{V}_7$ = BCD

⑤ = $V_0\overline{V}_1V_4\overline{V}_5\overline{V}_6V_7$ = DFMP + DUFM

⑥ = $V_0\overline{V}_1\overline{V}_4\overline{V}_5\overline{V}_6$ = MPF + MPY

⑦ = $V_0\overline{V}_1\overline{V}_2V_4\overline{V}_5\overline{V}_6\overline{V}_7$ = FMP + UFM

⑧ = $V_2V_4V_5\overline{V}_7\overline{V}_8$ = GTB

① FIXED COUNT/G1 = [DFDV + DFDI + (DFDV + DFDI) DU] (GD2)
= (1) (GD2)

② FIXED COUNT/G2 = [(FDI + FDV) + (DVF + DIV) + (FDI + FDV)
(DU)] (GD2) + (DFMP + DUFM + GTB) (GS)
= [(2 + 3)] (GD2) + [(5) ($\overline{\text{DU}}$) +(8)] (GS)

③ FIXED COUNT/G3 = [(MPF + MPY) + (MPF + MPY)DL] (GS)
= (6) ($\overline{\text{DU}}$) (GS)

④ FIXED COUNT/G4 = [(MPF + MPY) (DU) + (FMP + UFM)] (GS)
= [(6) (DU) + (7) ($\overline{\text{DU}}$)] (GS)

⑤ FIXED COUNT/G5 = [(FDI + FDV) + (FDI + FDV) (DU)
+ (DVF + DIV) + BCD] (GD2)
+ [(FMP + UFM) + (FMP + UFM) (DU)
+ (DFMP + DUFM) (DU) + GTB] (GS)
= [2 + 3 + 4] (GD2) + [7 + (5) (DU) + 8] (GS)

12.4.1.2 - contd

⑥

$$\begin{aligned}\text{FIXED COUNT/G6} &= [(\text{FMP} + \text{UFM}) + (\text{FMP} + \text{UFM})\ \text{DU}\\ &\quad + (\text{DFMP} + \text{DUFM})\ \text{DU} + (\text{MPF} + \text{MPY})\\ &\quad + (\text{MPF} + \text{MPY})\ \text{DU} + (\text{MPF} + \text{MPY})\ (\text{DL})]\ (\text{GS})\\ &\quad + [(\text{DVF} + \text{DIV}) + (\text{FDI} + \text{FDV})\\ &\quad + (\text{FDI} + \text{FDV})\ \text{DU} + \text{BCD}](\text{GD2})\\ &\quad + (\text{DFDV} + \text{DFDI})\ (\text{GD2})\\ &= [7 + 6 + 5\ (\text{DU})](\text{GS}) + (1+3+2+4)\ (\text{GD2})\end{aligned}$$

⑦

$$\begin{aligned}\text{FIXED COUNT/G7} &= [(\text{FMP} + \text{UFM}) + (\text{MPF} + \text{MPY}) + (\text{MPF} + \text{MPY})\ \text{DL}\\ &\quad + (\text{DFMP} + \text{DUFM})]\ (\text{GS})\\ &= [(7 + 6 + 5)\ (\overline{\text{DU}})]\ (\text{GS})\end{aligned}$$

⑧

$$\begin{aligned}\text{DU} &= \text{DIRECT MODIFIER FLIP-FLOP}\\ \text{\$ SET DUMF} &= [\text{\$DUMF}]_{\text{CF}}\\ \text{RESET DUMF} &= (\text{\$})\ (\text{GF})\ (\text{DUMF})_{\Delta} + \text{ABORT}\end{aligned}$$

12.4.2 G-COUNTER CONTROL DURING SHIFTS OR ALIGNS

12.4.2.1 Philosophy of Align Count Control During Floating Operations

Problem Statement

In performing floating-point addition and subtraction where the register bits are handled in parallel (to achieve computation speeds limited by circuit speeds and a practical degree of logic complexity), it is necessary to compare the magnitude of the exponents of addend and augend (for addition) or subtrahend and minuend (for subtraction) to determine how much the mantissa, associated with the smaller exponent, must be shifted to the right so as to achieve the digit alignment required before actually forming a sum. The exponent comparison consists of subtracting the exponent of the word from memory from the exponent of the word held ine an internal register. The resulting difference may be positive or negative. The sign determines which word is to be shifted right in the subsequent alignment. The magnitude determines how far right the shift is to proceed.

If either positive and/or negative magnitude representation is to be used in two's complement coding, then the magnitude decode logic will be twice as complex as that required for just positive number representation. This complexity may be avoided by converting any negative difference to its two's complement which is always positive and always has the identical magnitude. This alternative entails more time delay as the one's complement of the negative difference is added to unity (in the least bit position) to obtain the two's complement, and a carry may propagate the full length of the exponent adder. The following scheme avoids the delay of the two's complementation yet requires much less magnitude-detection logic than would be required by using both positive and negative differences in the two's complement coding.

12.4.2.1 - contd

Absolute-Value Detection

The initial exponent subtraction is performed without change. (In the 635, subtraction is performed by adding the one's complement [bit complement] of the subtrahend to the minuend after setting the least-significant carry bit.) If the difference is positive (indicated by the most significant bit of the adder output having a zero value), then the difference is transferred to the G-counter for storage (and subsequent countdown during alignment shifts). If the difference is negative, the one's complement of the adder is transferred to the G-counter except the sign bit (most significant bit) retains the unity value which denotes a negative difference.

The contents of the G-counter therefore will hold the required number of align counts if the most significant bit of the counter is zero (positive) and will hold the align count less one if the most significant counter bit is unity (negative). The align-shift logic thus must account for the difference in apparent count according to the sign bit of the G-counter. For many magnitude values, adjacent combinations represent the two possible representations in the counter, hence in some cases the shift logic is very little more complex than just that of all-positive representation.

12.4.2.1 - contd

Implementation

The G-counter has eight bits, so that excepting the sign bit 128 shifts can be counted. However, the mantissa register is limited to 72 bits so shift counts exceeding 71 simply result in the align mantissa becoming all zeros (including sign, hence the number zero). This mantissa-magnitude >71 detection takes place prior to transfer of difference to align counter, hence the >71 detection must work on conventional two's complement coding. A simple clearing of register substitutes for align-right shift shifting when the count exceeds 71. The following discussion assumes an align-shift count less than 72.

The 645 provides shifts of 16-, 4-, and 1-bit positions. Overshifting of 1-bit position is permitted, with subsequent correction, to achieve a significantly faster shifting operation. Thus, shift 16 is required whenever the align counter contains a number equivalent to 15 or more; shift 4 is required when the counter contains a number equivalent to 3 or more. All shifts of 16 must be performed before shifts of 4 and all shifts of 4 must be performed before shifts of 1. Overshift sets a Reverse flip-flop which signals that the next shift must be exactly 1-bit position in the opposite direction (under align which is normally a right shift, reverse calls for a left shift). The following table shows the required control combinations.

12.4.2.1 Table -- G-Counter Contents

Req'd Count Magnitude	G-Counter Contents																Control Operations and Comments
	Positive No.								Negative No.								
	G_0	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_0	G_1	G_2	G_3	G_4	G_5	G_6	G_7	
0	0	0	0	0	0	0	0	0									No negative representation of zero; do not shift
1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	Shift right one, set counter to zero
2	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	Shift right one, decrement counter by one
3	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	Shift right four, set Reverse, set counter to zero
4	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	Shift right four, set counter to zero
5	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	Shift right four, decrement counter by four
⋮	⋮	⋮	⋮	⋮					⋮	⋮	⋮	⋮					⋮ ⋮
13	0	0	0	0	1	1	0	1	1	0	0	0	1	1	0	0	Shift right four, decrement counter by four
14	0	0	0	0	1	1	1	0	1	0	0	0	1	1	0	1	Shift right four, decrement counter by four
15	0	0	0	0	1	1	1	1	1	0	0	0	1	1	1	0	Shift right 16, set Reverse, set counter to zero
16	0	0	0	1	0	0	0	0	1	0	0	0	1	1	1	1	Shift right 16, set counter to zero
17	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	Shift right 16, decrement counter by 16
⋮	⋮								⋮								⋮ ⋮
71	0	1	0	0	0	1	1	1	1	1	0	0	0	1	1	0	Shift right 16, decrement counter by 16
72	0	1	0	0	1	0	0	0	1	1	0	0	0	1	1	1	Shift counts of 72 or greater will not be put into counter for purposes of mantissa alignments.

12.4.2.1 - contd

Control Logic Resulting from Hybrid Coded Table

Shift 16 = $G_1+G_2+G_3+G_4 \cdot G_5 \cdot G_6(G_0+G_7)$ = SH16

Shift 4 = $(\overline{SH16})(G_4+G_5+G_6[G_0+G_7])$ = SH4

Shift 1 = $(\overline{SH16})(\overline{SH4})(G_0+G_6+G_7)$ = SH1

Set Reverse = $([SH16]\ \overline{G}_1\overline{G}_2\overline{G}_3 + [SH4]\overline{G}_4\overline{G}_5)(\overline{G}_0+\overline{G}_7)$

Clear G = (Last Align Shift) + (|Exponent Diff|>71)

Last Align
Shift = (G=16) + (G=4) + (G=1) + (G=-113) + G=-125)
+ (G=-128) + REV

With the exception of the "Last Align Shift" signal, the term "$\overline{G}_0 + \overline{G}_7$" or the term "$G_0 + G_7$" constitute the main difference in logical complexity between the requirements of this hybrid code scheme and an all-positive number representation (as would be obtained by two's complementing negative counts). This demonstrates the economy of this code scheme with this particular shifting scheme. The justification would be weaker if the over-shift and reverse scheme were not used. The negative count representation if read directly will always be $2^{n-1} + 1 - N$ where n is the number of bits in the counter and N is the absolute (positive) value of the shift count. Thus, exponent differences of (-16), (-4), and (-1) will appear in the counter as (-113), (-124), and (-128) respectively.

NOTE: The shift 16, shift 4, and shift 1 logic developed above also is applicable to shift instructions.

12.4.2.2 Equations for Transfer of Align Count from ES- to G-Counter

E -D = Positive (E-register minus D-register)

1) ¢ES_{0-7}/G_{0-7} = $(GE)(\overline{ES_0})(\overline{/ES/\geq 72})$

2) ¢ SET G_{0-7}

   a) ¢ SET G_0 = (¢ES_{0-7}/G_{0-7})(ES_0)
   b) ¢ SET G_1 = (")(ES_1)
   c) ¢ SET G_2 = (")(ES_2)
   d) ¢ SET G_3 = (")(ES_3)
   e) ¢ SET G_4 = (")(ES_4)
   f) ¢ SET G_5 = (")(ES_5)
   g) ¢ SET G_6 = (")(ES_6)
   h) ¢ SET G_7 = (")(ES_7)

E -D = Negative

1) ¢$\overline{ES}_{0-7}/G_{0-7}$ = $(GE)(ES_0)(\overline{/ES/\geq 72})$

2) ¢ SET G_{0-7}

   a) ¢ SET G_0 = ($\overline{¢ES}_{0-7}/G_{0-7}$)($ES_0$)
   b) ¢ SET G_1 = (")($\overline{ES}_1$)
   c) ¢ SET G_2 = (")($\overline{ES}_2$)
   d) ¢ SET G_3 = (")($\overline{ES}_3$)
   e) ¢ SET G_4 = (")($\overline{ES}_4$)
   f) ¢ SET G_5 = (")($\overline{ES}_5$)
   g) ¢ SET G_6 = (")($\overline{ES}_6$)
   h) ¢ SET G_7 = (")($\overline{ES}_7$)

12.4.2.3 Equations for Transferring the Number of Shift Counts to G-Counter

1) $¢M_{11-17}/G_{1-7} = (GS)\ (ALL\ SHIFTS)$

$= (GS)\ (V_0V_1V_2V_4)\ (\overline{GTB})$

$= (GS)\ (V_0V_1V_2V_4)\ (\overline{V_4\overline{V}_7\overline{V}_8})$

2) $¢\ SET\ G_{1-7}$

a) $¢\ SET\ G_1 = (¢M_{11-17}/G_{1-7})(M_{11})$

b) $¢\ SET\ G_2 = ($ " $)(M_{12})$

c) $¢\ SET\ G_3 = ($ " $)(M_{13})$

d) $¢\ SET\ G_4 = ($ " $)(M_{14})$

e) $¢\ SET\ G_5 = ($ " $)(M_{15})$

f) $¢\ SET\ G_6 = ($ " $)(M_{16})$

g) $¢\ SET\ G_7 = ($ " $)(M_{17})$

12.4.2.4 G-Counter Decrement During Shifts or Align

1) The execution of the number of shifts specified by the G-counter during all shift or align operations is on a 16, 4, and 1 sequential basis.

The G-counter is decremented by 16, 4, and 1 until the number of shift counts is expended. The controls for this shift order are:

a) GSH_{16} = (SHIFT 16)[GA + GO (SHIFTS)]

b) GSH_4 = SHIFT 4 [GA + GO (SHIFTS)]

c) GSH_1 = SHIFT 1 [GA + GO (SHIFTS)]

2) Counter logic during three counter decrement control.

a) SET G_7 = ($G_6\bar{G}_7$)

RESET G_7 = (G_7)

b) SET G_6 = 0

RESET G_6 = ($G_6\bar{G}_7$)

c) SET G_5 = ($G_4\bar{G}_5$)

RESET G_5 = (G_5)

d) SET G_4 = 0

RESET G_4 = $G_4\bar{G}_5$

e) SET G_3 = ($\bar{G}_3$)

RESET G_3 = (G_3)

f) SET G_2 = ($G_1\bar{G}_2\bar{G}_3$)

RESET G_2 = ($G_2\bar{G}_3$))

n						
G_1	G_2	G_3	G_4	G_5	G_6	G_7
1	1	1	1	1	1	1
1	1	0	1	0	1	0
1	0	1	0	1	0	1
1	0	0	0	0	0	0
0	1	1				
0	1	0				
0	0	1				
0	0	0				

n + 1						
G_1	G_2	G_3	G_4	G_5	G_6	G_7
1	1	0	1	0	1	0
1	0	1	0	1	0	1
1	0	0	0	0	0	0
0	1	1				
0	1	0				
0	0	1				
0	0	0				

12.4.2.4 - contd

g) SET $G_1 = 0$

RESET $G_1 = (G_1\overline{G}_2\overline{G}_3)$

3) "Clear" G-counter during shifts, align, or normalize.

a) The G-counter must clear to all zeros under the following conditions:

(1) $G = \pm 1 + G = \pm 4 + G = \pm 16$ For Shifts
or or
(2) $G = \pm 3 + G = \pm 15$ Aligns

Each of these counts indicates a last cycle shift forthcoming and will require the setting of G to zero.

(3) $GF + (GS)(\overline{GF})$ For initialize

b) "Clear" equation:

$$\text{Clear} = \{[(G = \pm 1) + (G = \pm 4) + (G = \pm 16)] + [(G = \pm 3) + (G = \pm 15)]\}$$

$$\{GA + [\text{ALL SHIFTS OPS}][GO]\} + GF \pm GS\ \overline{GF}$$

c) To insure that all ¢S/R Gi control points under control of GSH1, GSH4, or GSH16 are off when shift counts of 1, 4, 16, 3, or 15 are detected, the term clear is included in these control points as an inhibit.

12.4.2.4 - contd

4) ¢ SET/RESET Gi (under decrement by 16, 4, 1 control)

a) ¢ SET G_7 = (GSH1) ($\overline{\text{CLEAR}}$) ($G_6\overline{G}_7$)

¢ RESET G_7 = (GSH1) ($\overline{\text{CLEAR}}$) (G_7)

b) ¢ SET G_6 = 0

¢ RESET G_6 = (GSH1) ($\overline{\text{CLEAR}}$) ($G_6\overline{G}_7$)

c) ¢ SET G_5 = (GSH4) ($\overline{\text{CLEAR}}$) ($G_4\overline{G}_5$)

¢ RESET G_5 = (GSH4) ($\overline{\text{CLEAR}}$) (G_5)

d) ¢ SET G_4 = 0

¢ RESET G_4 = (GSH4) ($\overline{\text{CLEAR}}$) ($G_4\overline{G}_5$)

e) ¢ SET G_3 = (GSH16) ($\overline{\text{CLEAR}}$) ($\overline{G}_3$)

f) ¢ SET G_2 = (GSH16) ($\overline{\text{CLEAR}}$) ($G_1\overline{G}_2\overline{G}_3$)

¢ RESET G_2 = (GSH16) ($\overline{\text{CLEAR}}$) ($G_2\overline{G}_3$)

g) ¢ SET G_1 = 0

¢ RESET G_1 = (GSH16) ($\overline{\text{CLEAR}}$) ($G_1\overline{G}_2\overline{G}_3$)

12.4.3 G-COUNTER CONTROL DURING NORMALIZE

1) When normalization is required during floating operations, the number of shifts required to normalize a mantissa is stored in the G-counter.

   During GN the G-counter is incremented by factors of 16, 4, and 1. These increments are determined by the conditions of the NSL1, NSL4, and NSL16 detectors where

   a) NSL16 = Normalize Left 16-Bit Positions (G incremented by 16)

   b) NSL4 = Normalize Left 4-Bit Positions (G incremented by 4)

   c) NSL1 = Normalize Left 1-Bit Position (G incremented by 1)

2) ¢ SET/RESET Gi during normalization

   a) SET $G_7 = (NSL_1)(\overline{G}_7)$

      RESET $G_7 = (NSL_1)(G_7)$

   b) SET $G_6 = (NSL_1)(G_7)$

      RESET $G_6 = 0$

   c) SET $G_5 = (NSL_4)(\overline{G}_5)$

      RESET $G_5 = (NSL_4)(G_5)$

   d) SET $G_4 = (NSL_4)(G_5)$

      RESET $G_4 = 0$

12.4.3 - Contd

e) SET $G_3 = (NSL_{16})(\overline{G}_3)$

RESET $G_3 = (NSL_{16})(G_3)$

f) ¢ SET $G_2 = (NSL_{16})(\overline{G}_2 G_3)$

¢ RESET $G_2 = (NSL_{16})(G_2 G_3)$

g) ¢ SET $G_1 = (NSL_{16})(G_2 G_3)$

¢ RESET $G_1 = 0$

12.4.4 G-COUNTER DATA INPUT CONTROL

12.4.4.1 General

Data input to G fall into two general categories:

1) Fixed count data
    - a) ES, $\overline{ES}$/G (align)
    - b) M/G (shifts)
    - c) Number of multi-op subcycles (MPLY, DVD, etc.)

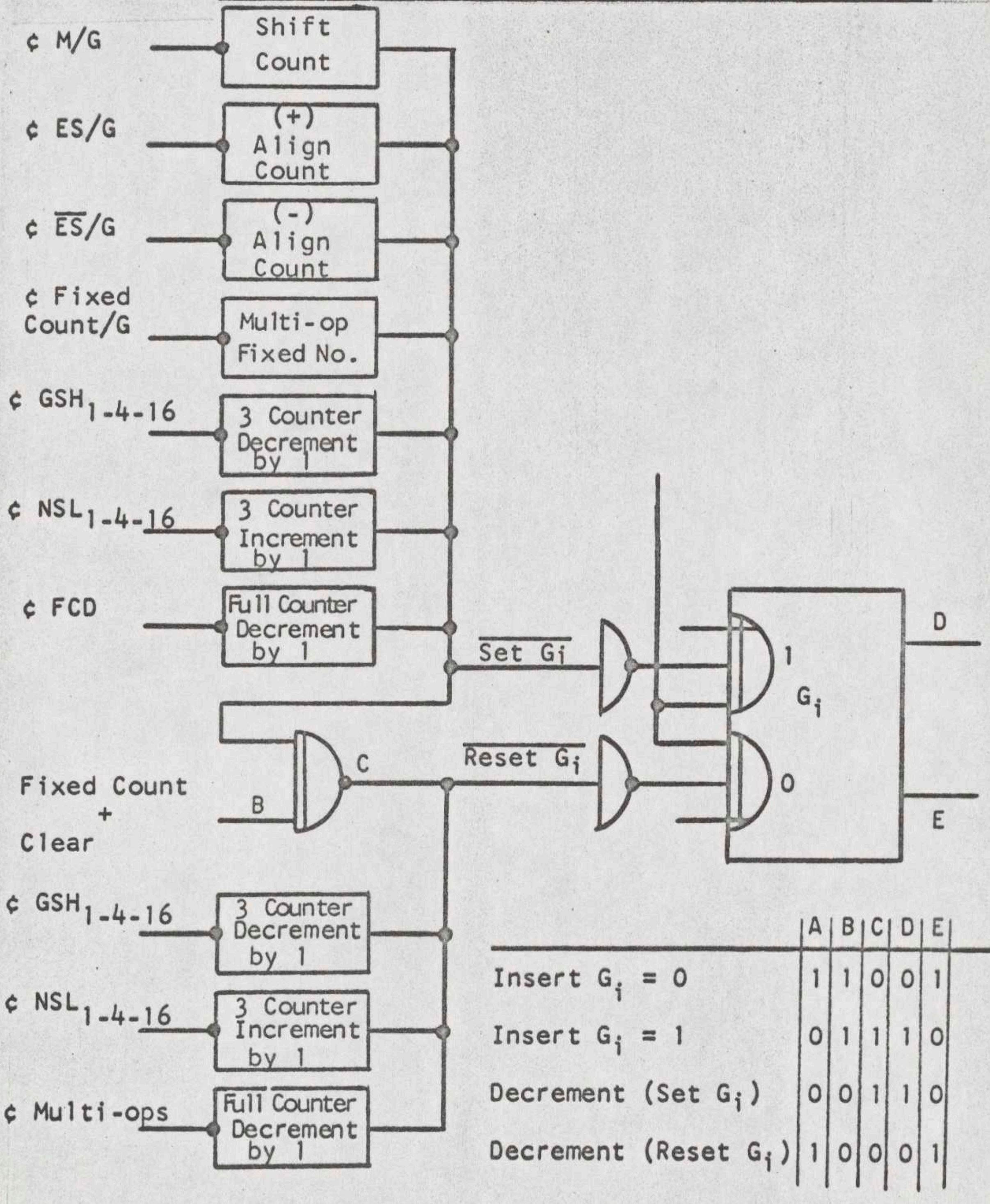
2) Counter control logic during shifts or aligns
    - a) Three counters-sequentially decremented by 1. (shifts or aligns)
    - b) Three counters-sequentially incremented by 1. (normalize)
    - c) One counter decremented by 1. (multi-ops)

In the case of category (1), it is desirable to have complementary data inputs to the Set/Reset inputs to each register flip-flop in order to force each flip-flop to the desired state regardless of its previous state. Counter initialization to "all 0's" can be included in category (1) (fixed count = 0). (Except when a fixed count is being inserted.)

Category (2) require separate and individually controllable Set/Reset inputs due to the register's counter function.

Conditional complementary controls for each G-counter bit is the method used to insure the counter will be forced to its proper state during fixed count insertion or when G must be cleared.

Conditional Complementary Control Block Diagram


¢ M/G
Shift Count
¢ ES/G
(+) Align Count
¢ $\overline{ES}$/G
(-) Align Count
¢ Fixed Count/G
Multi-op Fixed No.
¢ GSH_{1-4-16}
3 Counter Decrement by 1
¢ NSL_{1-4-16}
3 Counter Increment by 1
¢ FCD
Full Counter Decrement by 1
$\overline{\text{Set } G_i}$
$\overline{\text{Reset } G_i}$
Fixed Count + Clear
B
C
1
G_i
0
D
E
¢ GSH_{1-4-16}
3 Counter Decrement by 1
¢ NSL_{1-4-16}
3 Counter Increment by 1
¢ Multi-ops
Full Counter Decrement by 1

	A	B	C	D	E
Insert G_i = 0	1	1	0	0	1
Insert G_i = 1	0	1	1	1	0
Decrement (Set G_i)	0	0	1	1	0
Decrement (Reset G_i)	1	0	0	0	1

Fig. 12.4.4.1

12.4.4.2 Conditional Complementary Set/Reset (CCSR) Control Requirements

Complementary Set/Reset inputs to G_{0-7} are required when:

1) A specific number of minor cycles is being inserted or

2) The counter is to be reset to zero.

The insertion of a specified count into G is required when any one of the following control points is on:

1) ¢ ES/G (align count for E-D = positive difference)

2) ¢ $\overline{ES}$/G (align count for E-D = negative difference)

3) ¢ M_{11-17}/G_{1-7} (shift count)

4) Fixed Count/G (number of minor cycles during multiply, divide, BCD, or GTB)

The counter is to be reset to zero under the following conditions:

1) When the count in G during a Shift instruction or during mantissa alignment (GA) indicates 1, 3, 4, 15, or 16 shifts or alignments remain.

2) When G = 1 during multiply, divide, BCD, or GTB.

3) Unconditionally reset during (GS) ($\overline{GE}$) ($) and during (GF) ($).

12.5 REPEAT COUNTER DESCRIPTION

During any repeat operation (RPT, RPL, RPD) the OU will count the number of times the specified operation is repeated. Bits 0 through 7 of index register X0 form an 8-bit binary counter capable of counting from any number up to 255 down to 0 and on around again. During the execution of the repeat instruction itself this counter can be set to a specified number from 0 to 255, or left unaltered depending upon whether bit 10 of the repeat instruction word is a one or a zero. In either case it will be decremented each time the repeated operation is executed. This will continue until a terminate condition is met or the count reaches zero (tally runout). When a repeat operation terminates, the difference between the original count and the final count will be the number of times the operation was executed. If the original count was zero, the first operation will cause it to become all ones (255) and it will continue to count down until a terminate condition is met. Therefore, the maximum number of times an instruction can be repeated is 256.

When a repeat instruction is received from memory, it will be strobed into M_{0-35}, as well as into the instruction register. During GS it will be transferred to H_{0-35}. During GF the transfer path to X0 will be set up, and if H_{10} is a one, the contents of H_{0-17} will be strobed into $X0_{0-17}$. If the instruction is a RPT or RPL, the SENSE TERM flip-flop will be set by $ LAST. During a RPT or RPL operation the instruction to be repeated is sent to the OU. At that time the logic to decrement $X0_{0-7}$ will be enabled. Each time GS is reset, $X0_{0-7}$ will be strobed decreasing the count by one. Terminate conditions will not be checked until just prior to $ LAST of the repeated instruction. Therefore, the instruction will be executed at least once. If $X0_{0-7}$ was initially set to all zeros (repeat 256 times) it will be decremented to all ones prior to the Tally Runout check.

12.5 - Contd

During a RPD operation a pair of instructions will be repeated. When the first instruction is sent to the OU, the decrement logic will be enabled, however, SENSE TERM will not be set until $ LAST of that instruction, and it will toggle every $ LAST from then on. Therefore $X0_{0-7}$ will only be strobed at the end of the GS cycle for the second instruction. Likewise the check for Tally Runout is only made at the end of the second instruction. If $X0_{0-7}$ equals zero at $ LAST time, TALLY RUNOUT IND will be set and ¢ SET TRMF will be sent to the Control Unit to signal that the repeat mode termination procedure should be initiated.

12.6 INTERVAL TIMER DESCRIPTION

12.6.1 GENERAL

The Interval Timer (Timer Register) is a 24-bit binary counter which provides a program interrupt at the end of a variable interval. The counter can be set to a maximum of approximately 4 minutes total elapsed time via the LDT instruction. The decrement rate is 64 kc (15.625 microseconds) and is generated from a crystal controlled oscillator in the Control Unit.

12.6.2 LDT

The Processor must be in Master mode when executing the LDT instruction in order to load the Timer Register. The execution of this instruction in the Slave mode will have no effect upon the register. The load path to the timer is from the S-adder. The condition [GF] [LDT] [Master/Slave Ind = 1] will collector trigger the timer with the specified time interval. During this duration the 64 kc strobe will have no effect. Decrementing of the time interval will commence between 0 microsecond and 15.625 microseconds after the load is completed.

12.6.3 TIMER RUNOUT

Timer Runout is anticipated by sensing the condition (TIMER = 0 —— 01). This level generates a pulse ($ TIMER = 0) which is transmitted to the Base Frame setting the Timer Runout Fault flip-flop (TROF). The timer continues to count. If the Processor is in Master mode, TROF will remain set until the Slave mode is entered at which time the fault will be recognized and the flip-flop reset. The fault is immediately recognized if the Processor is in Slave at the time of runout.

12.6.4 STT -- SREG

The Timer Register is stored via the STT or SREG instructions. The store path is via the ZAQ bus onto the Data Out lines and is controlled as follows:

¢ $TIMER/ZAQ_{0-23}$ = (SREG)(GF)(GRU = 001) + (STT)(GC)

¢ ZAQ_{0-35}/DO_{0-35} = (same as above)

In order to insure that Timer Register is not in the process of trickle-down from the previous 64 kc pulse, the oscillator is turned off upon entering the STT or SREG instructions.

12.7 THE INDICATORS

The indicators may be regarded as individual bit positions in a half-word Indicator Register (IR).

An indicator is set to ON or OFF by certain events in the Processor, or by certain instructions. The ON state corresponds to a binary ONE in the respective bit position of IR; the OFF state corresponds to a ZERO.

The description of each machine instruction in EPS Section I includes a statement about the indicators that may be affected by the instruction, and the condition under which a setting of the indicators to a specific state occurs. If the conditions stated are not satisfied, the status of this indicator remains unchanged.

The Indicator instruction set comprises instructions (LDI, STI, STCI, RTD, RCU, SCU, RET) which transfer data between the lower half of a storage location and the Indicator Register. The following table lists the indicators that have been implemented and their relation to the bit positions of the lower half of a memory location.

12.7 - Contd

Bit Position	Indicator
18	Zero
19	Negative
20	Carry
21	Overflow
22	Exponent Overflow
23	Exponent Underflow
24	Overflow Mask
25	Tally Run-Out
26	Parity Error
27	Parity Mask
28	Absolute Mode
29	Must be zero
30	
31	
32	
33	
34	
35	

The indicators may be divided into three basic groups as follows:

a) Program Set

Zero
Negative
Carry
Overflow
Exponent Overflow
Exponent Underflow
Tally Runout

12.7 - Contd

b) Instruction Set

Overflow Mask
Parity Mask
Absolute Mode

c) Hardware Set

Parity

The Indicators in Group A may be subdivided into two groups:

1) Fault Trip Indicators

Overflow
Exponent Overflow
Exponent Underflow

2) Non Fault Trap Indicators

Zero
Negative
Carry
Tally Runout

The Indicators are not actually contained in a register, but are interspersed as individual flip-flops throughout the Arithmetic Frame in the Processor, in close proximity to the logic associated with the setting and resetting of them. There is the exception, the Absolute Mode indicator is in the Control Frame.

The following paragraphs contain descriptions of the control of each individual indicator.

12.7.1 ZERO INDICATOR

The Zero Indicator is affected by instructions that change the contents of a Processor register (A, Q, AQ, Xn, TR) or adder, and by comparison instructions.

Generally, the indicator is set ON when on completion of an instruction, the new contents of the affected register or adder contains all binary ZERO's, otherwise the indicator is set OFF. An exception to this is as follows: If, during any divide instruction it is discovered that the divisor is zero, the instruction is aborted and the zero indicator set. This indicator may be tested by TZE and TNZ instructions. The status of the indicator is not affected by these instructions.

12.7.2 NEGATIVE INDICATOR

The Negative Indicator is affected by instructions that change the contents of a Processor register, (A, Q, AQ, Xn, TR) or adder, and by comparison instructions.

Generally, the indicator is set ON when on completion of an instruction, the new contents of bit position #0 of this register or adder is a binary ONE; otherwise it is set OFF. Two exceptions to this are as follows: If any divide instruction is aborted for other than divisor equals zero, the sign indicator is set to indicate an illegal divide attempt. This indicator may be tested by the TPL, TMI instructions. The status of the indicator is not affected by these instructions. The second exception appears during the BCD instruction, during which it will be set or reset at the beginning of the operation, depending on the condition of bit #0 of the A-register.

12.7.3 CARRY INDICATOR

The Carry Indicator is affected by additions, subtractions, comparisons, and left shift instructions.

Generally, the indicator is set ON when a carry is generated out of bit position #0; otherwise it is set OFF. An exception to this is that on a left shift instruction if the bits shifted through position Zero are not all the same the carry indicator will be set. This is done to note the fact that the sign has changed and must be taken into account.

The Carry Indicator may be tested by the Transfer on Carry (TRC) and Transfer No Carry (TNC) instructions. The status of the indicator is not affected by these instructions.

12.7.4 OVERFLOW INDICATOR

The Overflow Indicator is affected by arithmetic instructions, Load Complement, Floating Normalize and Negate instructions. It is not affected by shifts, add or subtract logic or And/Or instructions. If the Overflow Indicator is set during an arithmetic operation, it is due to the result of the operation being too large a number to be represented in the associated register. If the Overflow Indicator is set during a load complement or negate type instruction, it is a result of the operand being maximally negative (10——0). The indicator may be tested by the TOV instruction, and will be reset as a result of the operation.

12.7.5 EXPONENT OVERFLOW INDICATOR

The Exponent Overflow Indicator is affected by floating point arithmetic operations, using the EAQ registers or the E register only. It is also affected by floating normalize and floating negate instructions. The indicator is set ON when the exponent of the result is larger than +127 which is the upper limit of the exponent range.

12.7.5 - contd

Since it is not set "OFF" otherwise, the Exponent Overflow Indicator reports any exponent overflow that has happened since it was last set OFF by certain instructions (see LDI, RET, TEO).

12.7.6 EXPONENT UNDERFLOW INDICATOR

The Exponent Underflow Indicator is affected by floating point arithmetic operations, using the EAQ registers or the E register only. It is also affected by Floating Negate and Floating Normalize instructions. The indicator is set ON when the exponent of the result is smaller than -128 which is the lower limit of the exponent range.

Since it is not set "OFF" otherwise, the Exponent Underflow Indicator reports any exponent underflow that has happened since it was last set OFF by certain instructions (see LDI, RET, TEU).

12.7.7 OVERFLOW MASK INDICATOR

The Overflow Mask Indicator can be set ON or OFF only by the instructions LDI, RET, RTD, and RCU.

When the Overflow Mask Indicator is ON, the setting of the Overflow Indicator, Exponent Overflow Indicator, or Exponent Underflow Indicator does not cause an Overflow Fault Trap to occur. When the Overflow Mask Indicator is OFF, such a trap will occur if any one of the three Overflow Indicators is set.

Clearing of the Overflow Mask Indicator to the "unmask" state will not generate a fault from a previously set Overflow Indicator, Exponent Overflow Indicator, or Exponent Underflow Indicator. The status of the Overflow Mask Indicator does not affect the setting, testing, or storing of the Overflow Indicator, Exponent Overflow Indicator, or Exponent Underflow Indicator.

12.7.8 TALLY RUNOUT INDICATOR

The Tally Runout Indicator is affected by the repeat instructions (RPT, RPD, RPL), and instructions using IT modification for all tally designators except I and F. If the Tally Runout Indicator is set on completion of a Repeat Type instruction this indicates the repeat sequence was terminated due to a tally count of zero. There is an exception to this; note that in the case of an RPL instruction, a link address of zero also sets the Tally Runout Indicator. If the indicator is off on completion of a repeat type instruction, this means the sequence was terminated due to a specified terminate condition being met. The indicator will also be set if the tally goes to zero during an instruction using IT modification. In this case, the indicator will not be set until the tally modified instruction has been executed. The reason for this is to prevent a transfer of Tally Runout Indicator Off instruction with an IT modification from transferring due to tally runout caused by the instruction itself. This indicator may be tested by the transfer on Tally Runout Indicator Off Instruction. The status of the indicator is not changed by the instruction.

12.7.9 PARITY ERROR INDICATOR

The Parity Error Indicator is set if a parity error is detected by memory under the following conditions:

a) If a parity error is detected in either word during an instruction Fetch of Y pair.

b) If a parity error is detected in either word of a double precision operand Fetch.

12.7.9 - contd

c) During a Single Precision Operand Fetch the memory actually pulls two words. If the address (Y) is odd, C(Y) and C(Y-1) are pulled. If the address is even C(Y) and C(Y+1) are pulled. The Processor is only notified i.e. Parity Indicator is set only if the parity error occurs in the C(Y). Contents of Y-1 or Y+1 are not reported to the Processor.

This indicator may be reset by the LDI, RCU, RTD, or RET instructions.

12.7.10 PARITY MASK INDICATOR

The Parity Mask Indicator can be set ON or OFF only by the instructions LDI, RET, RTD, and RCU.

When the Parity Mask Indicator is ON, the setting of the Parity Error Indicator does not cause a Parity Error Fault Trap to occur. When the Parity Mask Indicator is OFF, such a trap will occur if the Parity Error Indicator is set.

Clearing of the Parity Mask Indicator to the "unmask" state will not generate a fault from a previously set Parity Error Indicator. The status of the Parity Mask Indicator does not affect the setting, or storing of the Parity Error Indicator.

12.7.11 ABSOLUTE MODE INDICATOR

The Processor is always in Absolute Mode when this indicator is on. The Processor is in non-Absolute Mode when this indicator is off except for that time during which the Processor is performing an Execute Double as a result of a fault.

12.7.11 - contd

The Absolute Mode indicator is set if one of the instructions executed during a fault forced XED is any transfer instruction except TSS that causes a transfer of control. It can also be set by a RET or RTD instruction if the Processor is in Master Mode and bit 28 of the operand is a "1". It can also be set by a RCU instruction if bit 28 of Y+3 is a "1".

The Absolute Mode indicator is reset by the execution of a TSS instruction. It can also be reset by any transfer instruction that causes a transfer of control if bit 29 of the instruction word is a "1". It can also be reset by a RET, RTD, or RCU instruction if bit 28 is a "0".

12.8 SPECIAL CONTROL FLIP-FLOPS

12.8.1 ABORT - ABORT CYCLE

The ABORT flip-flop causes the OU to stop all operations and to reset all essential control flip-flops. An abort sequence can be started by any operation of the OU that will cause a fault, the termination of a repeated operation, or a fault sequence initiated by the CF.

$$\text{ABORT (set)} = (\$\ \text{SET ABORT})_{CF} + (\$\ \text{SET DVCK}) + (\$\ \text{SET OVFL FLT}) + (\$\ \text{SET TRMF})(\$\text{LAST})$$

$$\text{ABORT (reset)} = (\$\ \text{RESET ABORT})_{CF}$$

12.8.2 BCFF_1 - BYTE CONTROL

BCFF_1 when set, tells the OU that the current instruction has an IT modifier of CI or SC. It is used in conjunction with BMFF to enable the decode of the Byte Position Register to pick up the proper ZAQ/ZDO (that is, $\text{ZAQ}_{30-35}/\text{ZDO}_{18-23}$, etc.). When the CU detects an instruction calling for IT character modification it sends a pulse ($BPR) to the OU. This pulse strobes the character position into the Byte Position Register and sets BCFF. The OU will send or receive the proper character. BCFF will be reset at the end of the OU operation.

$$\text{BCFF}_1 \text{ (set)} = \$\text{BPR}$$

$$\text{BCFF}_1 \text{ (reset)} = \$(\text{GF}) + (\text{ABORT})$$

12.8.3 BCFF$_2$

BCFF$_2$ when set, tells the QU that the current instruction has an IT modifier of CI or SC, and it is of the nonstore type. It is used in conjunction with BMFF to enable the decode of the Byte Position Register to enable the proper control logic on the ZDI/M switches (that is, ZDI_{0-8}/M_{27-35}, ZDI_{18-23}/M_{30-35}, etc.). When the CU detects an instruction calling for IT character modification it sends a pulse ($BPR) to the QU. This pulse strobes the character position into the Byte Position Register and sets BCFF$_2$ if the instruction is a nonstore. The QU will receive the proper character.

$$BCFF_2 \text{ (set)} = \$BPR\ (\overline{STORES})$$

$$BCFF_2 \text{ (reset)} = \$(GS)(\overline{GF}) + (ABORT)_{CF}$$

12.8.4 BMFF - BYTE MODE

BMFF is only meaningful when BCFF is set. It is used to designate whether the character is 6-bit or 9-bit. If the indirect word from memory signifies a 9-bit character by having a 1 in bit 30, BMFF will be set by \$BPR. If bit 30 of the indirect word is a 0, indicating a 6-bit character, BMFF will remain reset. BMFF is reset at the end of the operation.

BMFF (set) = (ZDI_{30}) (\$BPR)

BMFF (reset) = \$(GF)

12.8.5 CMFF - COMPARE MAGNITUDE

During a fixed-point divide operation the absolute value of the dividend is compared against the absolute value of the divisor, and if it is equal to or greater than the divisor a Divide Check Fault occurs. This comparison is made during the first GO cycle. CMFF defines this cycle. It will be set for one cycle only.

CMFF (set) = (GD_2) $(\overline{BCD})$ (\$)

CMFF (reset) = (Fixed-Point Divide) (\$)

12.8.5 - contd

During a floating-point divide the absolute value of the dividend mantissa is compared against the absolute value of the divisor mantissa, and if it is equal to or greater than the divisor it will be shifted right and exponent incremented until it is smaller. The comparison and correction are done in the first GO cycle. However, each shift requires a complete cycle, therefore, the G-counter must be prevented from counting until the OU is ready to start the second GO cycle. Again CMFF set defines this period and prevents the decrementing of the G-counter. On the cycle that the dividend becomes less than the divisor CMFF will reset and the G-counter will be decremented.

$$\text{CMFF (set)} = (\text{GD}_2)\ (\overline{\text{BCD}})\ (\$)$$

$$\text{CMFF (reset)} = (\text{Floating-Point Divides})\ (\text{SC}_0)\ (\$)$$

12.8.6 DDF - DECODE DELAY FLAG

DDF is part of the control for $ START. In order to generate a $ START the OU must have had time to decode the op code in the O-register, the operand must be ready, and the previous instruction must be completed. DDF set indicates that the decode lines are stabilized. It is set for a sufficient duration after the recept of the op code to insure this. It is reset by $ START.

$$\text{DDF (set)} = (\$\ \text{Set GS})_{\Delta}$$

$$\text{DDF (reset)} = (\$\ \text{START})$$

12.8.7 DIC - DATA INHIBIT CONTROL

During any multiply operation the M-register is an active register. Therefore, it must be protected against spurious data on its inputs. Since there may be spurious data on the ZDI lines, the ZDI to M control lines must be turned off. This is accomplished with the DIC flip-flop. It will be set when an operand is received for a multiply operation and will be reset when that operation no longer requires the use of the M-register. DIC set will inhibit all ZDI to M control lines except those used for CI or SC operations. It is not necessary to inhibit the lines used for CI and SC because an IT modification cannot start until all operations have been completed, therefore these lines cannot be active during a multiply operation.

$$\text{DIC (set)} = (\text{MPLY})_{\text{ZOR}}\ (\$\text{YRY})$$

$$\text{DIC (reset)} = (G = 3)$$

12.8.8 DPFF - DOUBLE-PRECISION

During double-precision load type operations the M-register in the OU will receive two 36-bit words from memory. The DPFF flip-flop is one of the controls for the Data In lines and for the M-register Strobe. When set it will enable the ZDI/M control lines for the second half of a double-precision word and the proper $M pulses. When reset it will enable the ZDI/M control lines for the first half of a double-precision word or for a single-precision word and the proper $M pulses. It will be set upon receipt of the first half of a double-precision word and reset upon receipt of the second half of a double-precision word.

$$\text{DPFF (set)} = (\overline{\text{DPFF}})\ (\text{Double-Precision Ops})\ (\$\text{DRDY})$$

$$\text{DPFF (reset)} = (\overline{\text{Set DPFF}})\ (\$\text{DRDY})$$

12.8.9 DUMF - DIRECT MODIFIER

During a multiply operation the number of GO cycles to be executed is determined by the number of bits in the multiplier and the type of multiplication being done. If a Direct Upper modifier is called for, the multiplicand is premultiplied by 18 for single-precision multiplies. Consequently, the count inserted into the G-counter must be decreased proportionally. This is accomplished by DUMF. During any floating-point multiply DUMF set results in a count of 6 being inserted into the G-counter rather than a count of 15. During any fixed-point multiply DUMF set results in a count of 10 being inserted into the G-counter rather than 19. DUMF will be set by a Direct Upper signal from the CU and reset upon completion of the first cycle of the operation. DUMF will be set during any instruction that calls for a Direct Upper modifier, however, it will have no effect except on multiply instructions.

$$\text{DUMF (set)} = (\$\text{DUMF})_{\text{CU}}$$

$$\text{DUMF (reset)} = (\text{GS})(\overline{\text{GF}})(\text{DUMF})(\$)$$

12.8.10 EIFF - EXPONENT INCREMENT

During any divide operation the absolute value of the dividend in the N-register is compared against the absolute value of the divisor in the H-register by adding them together in the S adder. During the compare magnitude (CMFF) cycle, a carry out of the adder (SC_0) indicates that the dividend is equal to or greater than the divisor. If this is the case, a divide cannot take place. In fixed-point the divide will be aborted, but in floating-point the dividend mantissa in N can be shifted right until the absolute value is less than the divisor mantissa thus permitting the divide to take place. When this is required the exponent in the E-register must be incremented by one for each place the dividend is shifted to maintain the true value of the dividend. The incrementing of E is under control of EIFF.

During a floating-point divide the first CMFF cycle will add N and H. If N is equal to or greater than H a carry out of adder bit zero (SC_0) will result. SC_0 will prevent CMFF resetting, will set EIFF, and will shift N one bit position right. On the next CMFF cycle N and H will be added again and EIFF set will gate the E-register to the ES adder, will gate a +1 to the ES adder, and will strobe the result back into E. If SC_0 is false, CMFF and EIFF will reset and N will not be shifted. At this point the divide operation will commence.

During a fixed-point divide a CMFF cycle will add N and H. If N is equal to or greater than H, SC_0 will come true. EIFF will set, but the divide operation will abort and EIFF will be reset by ABORT.

$$\text{EIFF (set)} = (\text{CMFF})\ (SC_0)\ (\$)$$

$$\text{EIFF (reset)} = (\text{CMFF})\ (\overline{SC_0})\ (\$) + (\text{ABORT})_{CT}$$

12.8.11 FRD - REPEAT DOUBLE

If the CU is in the Repeat Double mode of operation FRD in the OU will set. FRD set will allow SENSE TERM to toggle every OU operation so that it will allow the detection of terminate conditions only on the odd instruction. It will enable the set of TALLY RUNOUT IND. If XO_{17} is a zero it will prevent the detection of an overflow. (See GE-645 EPS, Section I for details on terminate conditions for a RPD instruction.) FRD will be reset upon completion of the Repeat Double Mode of Operation.

FRD (set) = (RPD) ($ LAST)

FRD (reset) = (TRMF) ($ LAST)

12.8.12 FTL - REPEAT/REPEAT LINK

If the CU is in the Repeat or Repeat Link mode of operation FTL in the OU will set. FTL set will accomplish the same function as FRD except that SENSE TERM will set and stay set for the entire operation.

FTL (set) = (RPT + RPL) ($ LAST)

FTL (reset) = (TRMF) ($ LAST)

12.8.13 GRU3, GRU2, GRU1

The G-counter is physically located in the arithmetic frame. In order to speed up the decode of various G-counts in the RF a 3-bit Shift Register is employed (GRU_3 GRU_2, GRU_1). In any multiple GO cycle operation except shifts, GRU_3 will be set when the G-counter is stepped from 4 to 3. Each succeeding \$ pulse will shift the bit down one until the counter is cleared. These three flip-flops are used to decode the various G counts required by the register frame. This counter also is used by the Load Registers and Store Registers operations. It will determine which registers are to be loaded or stored during each sequence. The counter will be stepped at the end of each GF cycle. It will count as follows:

GRU_3	GRU_2	GRU_1	
0	0	0	load or store X0, X1, X2, and X3
1	0	0	load or store X4, X5, X6, and X7
0	1	0	load or store AQ
0	0	1	load or store E and Timer

$$GRU_3 \text{ (set)} = [(GO)\ (\overline{Shifts})\ (G{=}4) + (GF)\ (LREG + SREG)]\ [\$GRU]$$

$$GRU_3 \text{ (reset)} = (GRU_3)_\Delta\ (\$GRU)$$

$$GRU_2 \text{ (set)} = (GRU_3)_\Delta\ (\$GRU)$$

$$GRU_2 \text{ (reset)} = (GRU_2)_\Delta\ (\$GRU)$$

$$GRU_1 \text{ (set)} = (GRU_2)_\Delta\ (\$GRU)$$

$$GRU_1 \text{ (reset)} = (GRU_1)_\Delta\ (\$GRU)$$

$$\$GRU = [\overline{(GF)\ (LREG + SREG)}]\ \$$$

12.8.14 H_{72}

During all multiply operations the partial product is formed in the S-adder and shifted two places right when stored back in H (that is, S_{0-70} to H_{2-72}). Normally it is not necessary to remember what was shifted one place left and stored in N (that is, H_{1-72} to N_{0-71}). H_{72} holds the bit shifted from S_{70} for later insertion into N_{71}.

$$H_{72} \text{ (set)} = (S_{70}) (\$)$$
$$H_{72} \text{ (reset)} = (\overline{S_{70}}) (\$)$$

12.8.15 LAFF - LOGICAL ALIGN

During a floating-point subtract, compare, or compare magnitude instruction a peculiar problem can occur. Normally the mantissa alignment is accomplished by arithmetic right shifts. However, if the mantissa in memory is a maximum negative number (that is, sign bit a 1 and all other bits a 0) when it is ones complemented and stored in H the sign bit will be a 0 and all other bits a 1. If this mantissa is chosen to be aligned, because its exponent is less positive than the other exponent, it will be routed through the S-adder to N. K72 will be on converting the ones complement to a twos complement. This will result in the mantissa in N having sign bit A 1 and all other bits a 0 (that is, a maximum negative number). N therefore now contains a mantissa which overflowed when twos complemented. However, when the twos complement was formed in the S-adder this detected overflow condition set LAFF. LAFF is in the N shift control logic and in the decrement G-counter logic. It will cause the first shift of N to be a logical right shift of one place making N sign bit a zero, N bit one a 1 and all other bits a 0. Thus, the overflowed mantissa is corrected, the G-counter is decremented by one, and the align cycle(s) continue with normal arithmetic shifts.

$$\text{LAFF (set)} = (\text{GE}) (\text{OFL}) (\overline{\text{ES}_9}) (\$)$$

$$\text{LAFF (reset)} = [(\overline{\text{GE}}) + (\overline{\text{OFL}})] (\$)$$

12.8.16 LAST

During a Repeat operation an instruction or a pair of instructions will be repeated until the specified terminate conditions have been met. If indicators such as zero or sign are checked before the operation is complete, erroneous terminates could occur. LAST insures that terminate conditions will only be tested between the end of GF and $ LAST. LAST and SENSE TERM enable ¢ SET TRMF.

$$\text{LAST (set)} = (\text{GF})\ (\$)$$

$$\text{LAST (reset)} = \$\ \text{LAST}$$

12.8.17 N-1

Shift logic allows shifting N either direction in increments of 1, 4, or 16 bits. To speed up shifting it is permissible to overshift by one and then reverse shift by one. Any left shift whether program or normalization can result in this overshift. In order to prevent losing a bit an extra flip-flop (N_{-1}) holds the overshifted bit. It will be shifted back into N_0 on the reverse shift.

$$N_{-1}\ \text{(set)} = [(N_3)\ (\text{GSH4}) + (N_{15})\ (\text{GSH16})]\ (\$)$$

$$N_{-1}\ \text{(reset)} = [(\overline{N_3})\ (\text{GSH4}) + (\overline{N_{15}})\ (\text{GSH16})]\ (\$)$$

12.8.18 N36A

Any type of single-precision right shift can result in an overshift and reverse shift. It is possible to shift from the upper half of N into the lower half of N to prevent loss of information. However, this nullifies the zero detection. An additional flip-flop N36A handles this situation. N36A will hold the last bit shifted out of the upper portion of N and N36-71 will remain all zeros. If an overshift occurred a reverse shift of one is required. The content of N36A will be shifted into N_{35}.

$$\text{N36A (set)} = [(N_{20})(GSH_{16}) + (N_{32})(GSH_{4})]\ \$$$

$$\text{N36A (reset)} = [\overline{\text{¢ Set N36A}}]\ \$$$

12.8.19 N_{72}

Any right shift, whether program or alignment, can result in an overshift and reverse shift. In order to prevent losing the Overshifted bit an Extended N-register Flip-flop (N_{72}) holds the overshifted bit for reinsertion into N_{71} on the reverse shift.

$$N_{72}\ \text{(set)} = [(N_{68})(GSH4) + (N_{56})(GSH16)]\ (\$)$$

$$N_{72}\ \text{(reset)} = [(\overline{N_{68}})(GSH4) + (\overline{N_{56}})(GSH16)]\ (\$)$$

12.8.20 ORBF - OPERATION REGISTER BUSY

When an operation code is received in the OU it will be stored in the O-register. The decode for the GS cycle originates from the O-register. At the end of GS the op code (which is now contained in the C-register) will originate from the C-register, freeing the O-register for a new operation code. ORB defines that period of time when the O-register contains a valid op code. It is used to prevent the strobing of 0 and it enables the $ SET GS logic. It will be set when an op code is strobed into the O-register and reset at the end of the GS cycle for that op code.

ORBF (set) = $OR

ORBF (reset) = (GS) $(\overline{GF})$ ($)

12.8.21 ORYF - OPERATION DESIGNATOR READY

When the CU has an operation code ready for the OU it sends a pulse ($ORY) notifying the OU of that fact. This pulse will set ORYF, which will remember that an op code is on the ZOR lines. When the O-register becomes free $(\overline{ORB}=1)$, the condition $(\overline{ORB})$ (ORY) will generate a $OR pulse, which will strobe the new op code into the O-register, set ORBF, and reset ORYF.

ORYF (set) = ($ORY)

ORYF (reset) = ($OR)

12.8.22 OUBF - OPERATION UNIT BUSY

The setting of GS is dependent upon the condition of the OU. If the OU is not busy all that is required is the receipt of an op code. If the OU is busy then the setting of GS must wait until the GF cycle of the current operation or in some cases until the GF cycle has been completed. OUBF, when set, indicates that the OU is performing operations and it prevents ORBF, which is set by the receipt of an op code, from generating a \$ SET GS. When OUBF is reset the condition $(\overline{\text{OUB}})$ (ORB) will result in a \$ SET GS. OUBF is set by \$ START and will stay set as long as the OU is processing an instruction(s).

$$\text{OUBF (set)} = (\$\ \text{START})$$

$$\text{OUBF (reset)} = (\text{GF})\ (\overline{\text{GS}})\ (\$)$$

12.8.23 PASS AND PPS

During a GO cycle of a multiply operation the multiplicand in AQ is added to or subtracted from the partial product in H. The result (new partial product) is shifted two places right and stored back into H_2 - H_{71}. This would normally result in H_0 and H_1 being cleared to zero. There are conditions where either or both H_0 and H_1 must be set to one. Two flip-flops (PASS and PPS) are used to detect these conditions. Whenever the effective multiplier is zero or four the pass flip-flop is set.

$$\text{PASS (set)} = \{[(\text{MPLY})]\ [\text{GS} + \text{GO}(\text{G}>2)]$$
$$[\overline{M_{62}}\ \overline{M_{63}}\ \overline{\text{STFF}} + M_{62}\ M_{63}\ \text{STFF}]\}\ \$$$

$$\text{PASS (reset)} + [\overline{\text{¢ (PASS) set}}]\ \$$$

12.8.23 - contd

Due to the fact that the partial product in H is always shifted two places to the right its absolute value is always less than that of the multiplicand. Therefore the partial product sign will always be the same as the sign of the multiplicand if an add takes place and opposite if a subtract takes place. PPS indicates the sign of the partial product.

$$\text{PPS (set)} = [(\text{Sign of Icand})\ (\text{ADD}) + \overline{(\text{Sign of Icand})}\ (\text{Sub})]\ \$$$

$$\text{PPS (reset)} = [\overline{(\text{Sign of Icand})}\ (\text{ADD}) + (\text{Sign of Icand})\ (\text{Sub})]\ \$$$

The effect of these two flip-flops will be as follows:

a) If PASS is set, H_0 and H_1 will be set if the partial product is negative or H_0 and H_1 will be reset if the partial product is positive.

b) If PPS is set, H_0 will always be set.

c) If PPS is set and the current partial product is positive, H_1 will be set in the new partial product.

d) If a carry out of the S adder occurs, H_1 will be set regardless of whether PPS is set or reset.

e) In all other cases H_0 and H_1 will be reset.

12.8.24 QSF - QUOTIENT SIGN

During the GO cycles of a divide operation the OU will always add a negative divisor to a positive dividend producing a positive quotient. However, if the signs of the original divisor and dividend were different the quotient should be negative. During the first set up cycle of a divide (GD_1) the signs of the divisor and dividend are compared, if they are different QSF will set and remain set until the next operation resets it. During the next to last GO cycle of a divide the accumulated positive quotient in AQ is transferred to N. If QSF is reset, indicating that the quotient should be positive, a straight transfer will take place. If QSF is set, indicating that the quotient should be negative, the ones complement will be transferred to N and K_{72} will be set to change it to a twos complement on the next cycle.

$$\begin{aligned} \text{QSF (set)} &= (GD_1)\,[(M_0\,\overline{A_0} + \overline{M_0}\,A_0)\,(DVF + FDV \\ &\quad + DFDV + FDI + DFDI) + (M_0\,\overline{Q_0} \\ &\quad + \overline{M_0}\,Q_0)\,(DIV)]\ \$ \end{aligned}$$

$$\text{QSF (reset)} = (GS)\ \$ + ABORT$$

12.8.25 REV - REVERSE

During any operation that requires shifting N, such as alignment, normalization, or programmed shift, it is possible to overshift by one bit. When this happens a reverse shift of one place is required. On any shift cycle resulting in an overshift REV will set. REV being set dictates that the next movement of N will be one bit position in the direction opposite to the movement which resulted in the overshift.

$$\begin{aligned}
\text{REV (set)} = \{&(\text{GN})\ [(\text{N DET 4B})\ (\text{N4}) + (\text{N DET 4A})\\
&(\overline{\text{N4}}) + (\text{N DET 16B})\ (\text{N16}) +\\
&(\text{N DET 16A})(\overline{\text{N16}})] + [\text{GA+GO(SHIFTS)}]\\
&[(\text{G}=\pm 3) + (\text{G}=\pm 15)] + (\text{GO})(\text{OFL})\ [(\text{FAD})\\
&+ (\text{DFAD}) + (\text{FSB}) + (\text{DFSB}) + (\text{FNO})\\
&+ \text{FNEG})] + [(\text{GO})\\
&(\text{G}=1)(\text{S}_0 \oplus \text{S}_1)\ (\text{Flt}_{\text{ng}}.\text{Mplys}] + (\text{GO})\ (\text{OFLI})\\
&(\text{FNO})\}\ \$
\end{aligned}$$

$$\text{REV (reset)} = (\text{REV})_{\Delta}\ \$$$

12.8.26 SDY_1 and SDY_2 - STORE READY

During store operations the OU generates a \$SDY to inform the CU that the operand will be ready for storage by the time memory will be ready for it. SDY_1 will be set when a store instruction, other than a slow store, is strobed into the O-register and it will generate a \$ DSY upon completion of the preceding operation. SDY_2 will be set when a slow store instruction is strobed into the O-register and will allow (GS) $(\overline{GF})$ \$ to generate a \$DSY.

$$SDY_1\ (\text{set}) = (\text{Slow Stores})\ (\$OR)$$

$$SDY_1\ (\text{reset}) = (\overline{\text{Slow Stores}})\ (\$OR)$$

$$SDY_2\ (\text{set}) = [(\text{Fast Store})+(\text{DP Store})+(\text{RAR})](\$OR)$$

$$SDY_2\ (\text{reset}) = [(\overline{\text{Fast Store})+(\text{DP Store})+(\text{RAR}})](\$OR)$$

12.8.27 SENSE TERM

During the Repeat Mode of operation the OU determines when to end the sequence. The OU attempts to check the terminate conditions during the time from the end of GF to \$ LAST of each instruction. However, it can only do so if SENSE TERM is set. This control is necessary to prevent the detection of terminate conditions on the even instruction of an even/odd pair during Repeat Double operation. During a Repeat or Repeat Link operation SENSE TERM will be set by \$ LAST of the RPT or RPL instruction and stay set until Repeat Mode terminates. During a Repeat Double operation SENSE TERM will be set by \$ LAST of the first repeated instruction and reset by \$ LAST of the second repeated instruction. It will be toggled by every instruction after that. This will allow the detection of terminate conditions only on the odd instruction. SENSE TERM set will also do the following:

a) Enable the reset logic for FTL and FRD.

b) Enable the set logic for TALLY RUNOUT IND.

c) Inhibit the generation of \$SDY on store instructions if the terminate conditions have been met.

d) Enable the XO-register Repeat Counter Control logic.

$$\text{SENSE TERM (set)} = [(\text{FRD}+\text{RPT}+\text{RPL})\ (\overline{\text{SENSE TERM}})]\ (\$\ \text{LAST})$$

$$\text{SENSE TERM (reset)} = (\text{FRD} + \text{TRMF})\ (\text{STR})\ (\$\ \text{LAST})$$

12.8.28 STFF - STRING

During multiply operations the multiplier is looked at two bits at a time (M_{62} M_{63}) to form an effective multiplier for that cycle. This means that the effective multiplier can be 0, 1, 2, or 3. The OU can handle multipliers of 0, 1, or 2 directly, however, an effective multiplier of 3 requires subtracting the multiplicand from the partial product once and adding it four times. The adding four times is accomplished by increasing the next effective multiplier pair by one. This means that it is possible to have an effective multiplier of 0, 1, 2, 3, or 4. An effective multiplier of 4 is handled by making this effective multiplier act like a 0 multiplier and adding one to the next effective multiplier pair. The fact that one must be added to the next effective multiplier is remembered by the STFF flip-flop. It is used along with M_{62} M_{63} in the decoding of the functions to be performed during the GO cycles of a multiply instruction. STFF will be set whenever the effective multiplier is a 3 or a 4.

$$\begin{aligned}
\text{STFF (set)} &= \{(\text{Multiply})[\text{GS} + (\text{GO})\ (\text{G}>3)]\} \\
&\quad \{(M_{62})\ (M_{63})\ (\overline{\text{STFF}}) + (\overline{M_{62}}) \\
&\quad (M_{63})\ (\text{STFF})\}\ \$ \\
\text{STFF (reset)} &= \{\text{Multiply}\}\ \{[\text{GS} + \text{GO}\ (\text{G}>3)] \\
&\quad [(\overline{M_{62}})\ (\overline{M_{63}}) \\
&\quad (\overline{\text{STFF}}) + (\overline{M_{62}})\ (M_{63})\ (\text{STFF}) + (M_{62}) \\
&\quad (\overline{M_{63}})\ (\overline{\text{STFF}})] + (\text{GO})\ (\text{G} = 3)\}\ \$
\end{aligned}$$

12.8.29 YRY - OPERAND READY

One of the criteria for the generation of $ START is that the operand will be ready by the time it is required. YRY will be set when this criterion is met. If the operation is a load type, YRY will be set by $YRY, which is generated by a pulse from the CU when the operand is available. If it is a double-precision load, YRY will not set until the second half of the operand is available. Since the operand is immediately available for a store type operation YRY can be set shortly after the GS cycle is started.

YRY (set) = $YRY

= ($DRDY) ($\overline{\text{¢ SET DPFF}}$) + ($ SET DDF)

(Stores)

YRY (reset) = ($ START)

12.9 LAST OF 2, 3 OR n PULSE DETECTOR

In an asyncronous environment it is sometimes necessary to generate a pulse conditioned on more than one variable. The technique described below is used in the GE-645 for producing the strobes, $ ØR (places the operation code in the Ø-register and sets ØRB flip-flop), $ START (starts the timing pulse generator and the ØU operation), and $SDY (Store data ready).

The last pulse detector is to produce a pulse at the time the last of pulses A, B, C, ... n has set or reset a specified flag. This is advised by anding flags A, B,....n. At the time the last input to the AND gate (See fig. 12.9) achieves a "1" state the output goes to "0". This drop in voltage causes the pulse amplifier to generate a positive pulse. In order to generate a new pulse, one of the flags A, B, C,....n must change to the opposite state through a set or reset pulse. This causes the output to rise to "1". The detector is now ready to generate a new pulse.

12.9 - contd

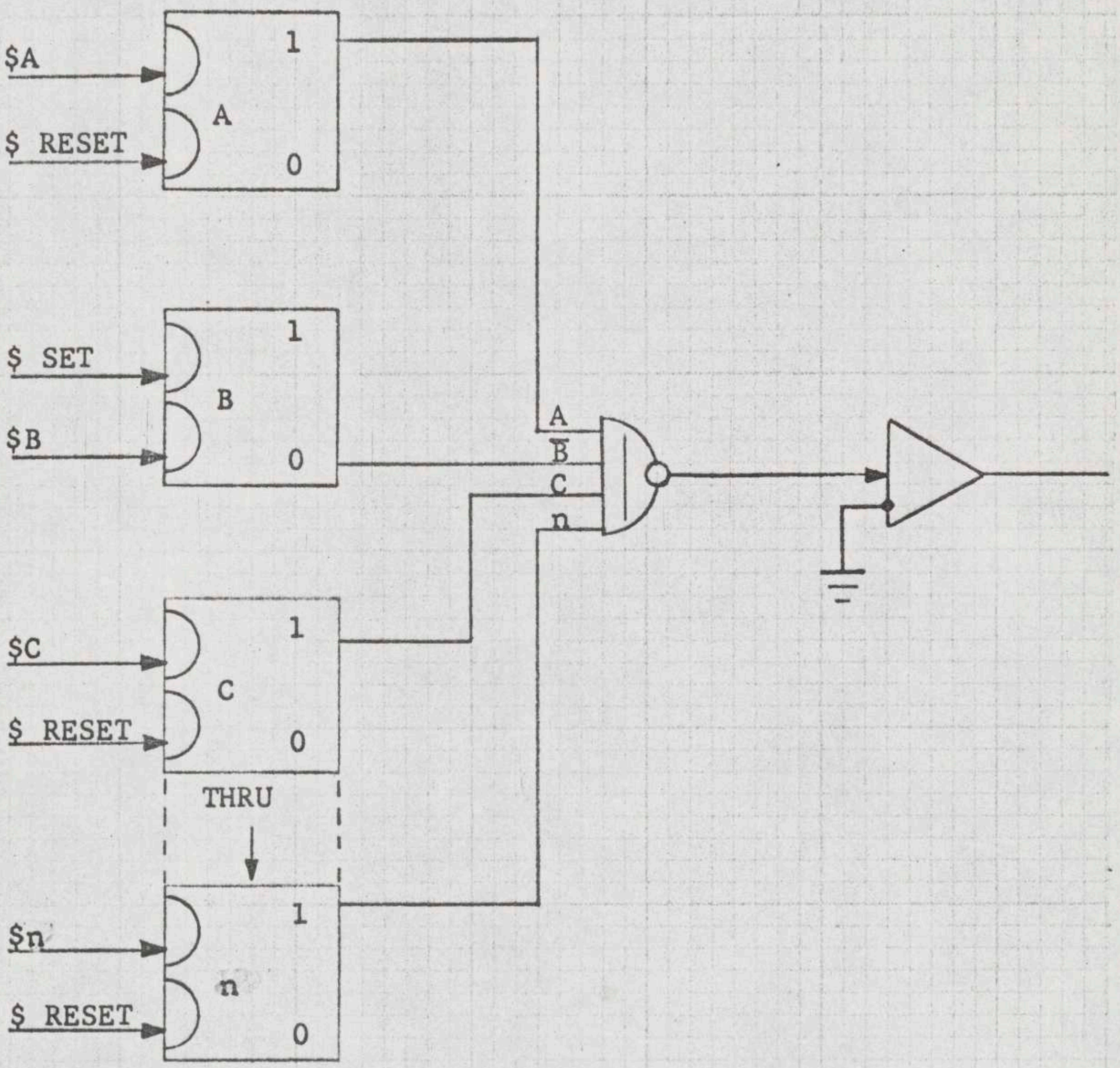

$A
$ RESET
A
1
0
$ SET
$B
B
1
0
$C
$ RESET
C
1
0
THRU
$n
$ RESET
n
1
0
A
$\overline{B}$
C
n

Fig. 12.9
Last of 2, 3 or n Pulse Detector, Logic Diagram

12.10 "S" ADDER

12.10.1 GENERAL

The Operations Unit S Adder is a two's complement parallel binary adder which forms the sum of two 72-bit operands contained in the holding registers N_{0-71} and H_{0-71}. Functionally the adder consists of 12 parallel channels each of which processes corresponding 6-bit bytes of the 72-bit operands held in the N- and H-registers. Each channel processes its assigned operand bytes simultaneously with all other channels so that any given byte resultant sum is available at the adder output at the same time as any other byte independent of its position in the 72-bit word. That is, the most significant byte sum (S_{0-5}) will be available at the output of channel 1 at the same time as S_{6-11} out of channel 2, S_{12-17} out of channel 3, etc.

The S Adder may be used to perform the functions of algebraic addition, inclusive OR, exclusive OR, or the logical product (AND) of the two operands contained in N and H. The operation to be performed is controlled by a 3-bit adder command register designated as the K-register. In addition a flip-flop K_{72} is used to add a one to the least significant bit of the adder. The adder command codes are:

K_1	K_2	K_3	
0	0	0	$H_{0-71} + N_{0-71}$
1	1	d	$H_{0-71} \vee N_{0-71}$
1	0	0	$H_{0-71} \textcircled{V} N_{0-71}$
1	0	1	$H_{0-71} \cdot N_{0-71}$

where "+" means arithmetic summation, "V" means inclusive OR summation, " $\textcircled{V}$ " means exclusive OR summation, and "·" means logical product (AND). Subtraction is performed by placing the logical complement of the subtrahend in H or N, setting K_{72}, and adding.

12.10.1 - contd

A functional block diagram of the S Adder is shown in Fig. 12.10.1. Each channel (except 11 and 12) consists of five functional blocks of logic, four of which are implemented as modules each on a single logic card. The five functional blocks of logic are:

1) Byte Carry Detector module designated as the L0M306A logic card.

2) Channel Carry Generator which is implemented with general purpose logic cards.

3) L0M303A 2-bit adder cards.

4) L0M304A 2-bit adder cards.

5) L0M303A 2-bit adder cards.

The time delay through any one 6-bit byte channel is 126 nanoseconds ±36 nanoseconds. Since all channels operate in parallel the total adder delay required to form the sum of any two numbers from 1 to 72 bits in length will be the same as the delay for longest channel.

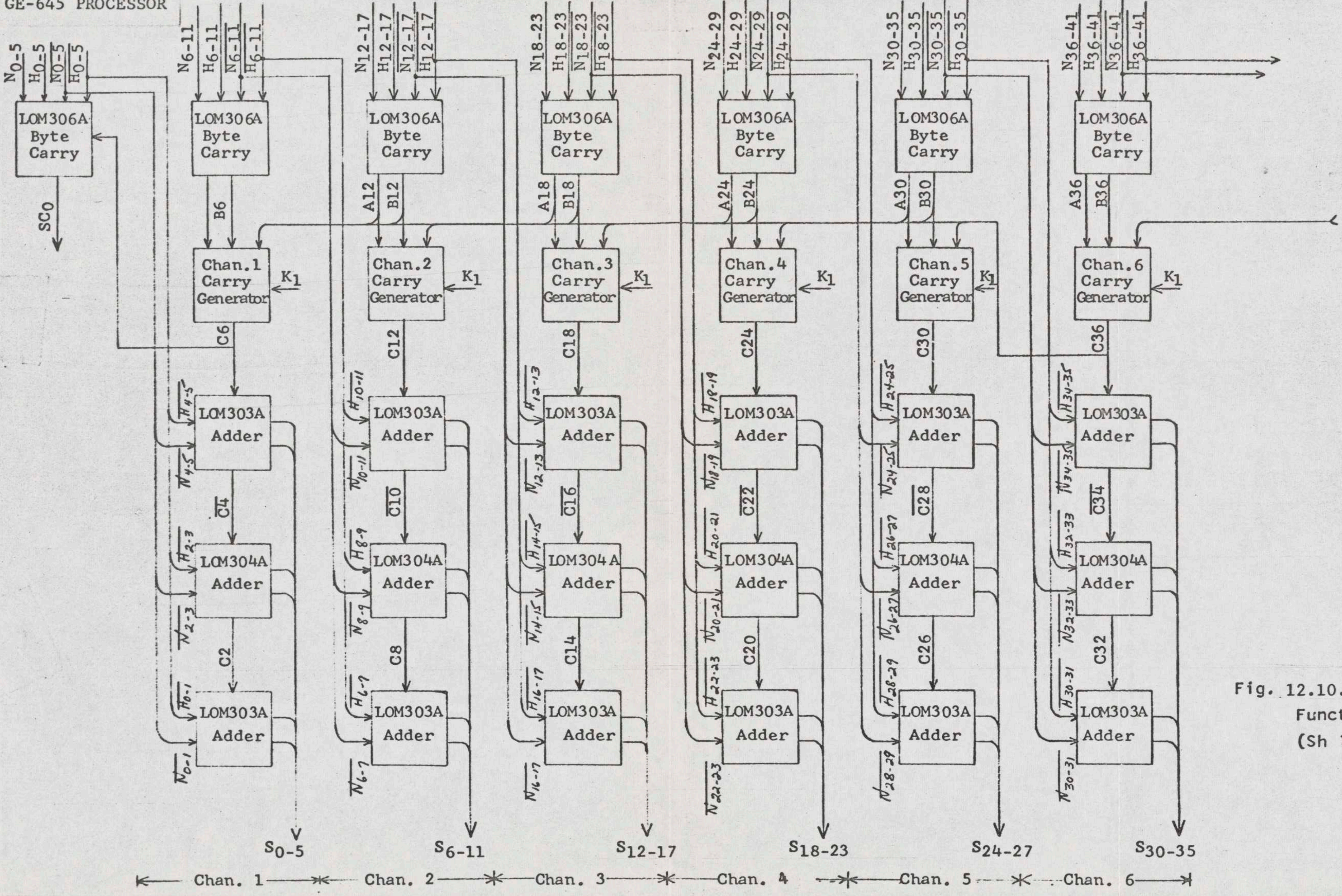

N_{0-5} H_{0-5} $\overline{N_{0-5}}$ $\overline{H_{0-5}}$
N_{6-11} H_{6-11} $\overline{N_{6-11}}$ $\overline{H_{6-11}}$
N_{12-17} H_{12-17} $\overline{N_{12-17}}$ $\overline{H_{12-17}}$
N_{18-23} H_{18-23} $\overline{N_{18-23}}$ $\overline{H_{18-23}}$
N_{24-29} H_{24-29} $\overline{N_{24-29}}$ $\overline{H_{24-29}}$
N_{30-35} H_{30-35} $\overline{N_{30-35}}$ $\overline{H_{30-35}}$
N_{36-41} H_{36-41} $\overline{N_{36-41}}$ $\overline{H_{36-41}}$
LOM306A Byte Carry
SC0
B6
A12 B12
A18 B18
A24 B24
A30 B30
A36 B36
Chan.1 Carry Generator
Chan.2 Carry Generator
Chan.3 Carry Generator
Chan.4 Carry Generator
Chan.5 Carry Generator
Chan.6 Carry Generator
K_1
C6
C12
C18
C24
C30
C36
LOM303A Adder
LOM304A Adder
$\overline{C4}$ $\overline{C10}$ $\overline{C16}$ $\overline{C22}$ $\overline{C28}$ $\overline{C34}$
C2 C8 C14 C20 C26 C32
$\overline{H_{4-5}}$ $\overline{N_{4-5}}$ $\overline{H_{2-3}}$ $\overline{N_{2-3}}$ $\overline{H_{0-1}}$ $\overline{N_{0-1}}$
$\overline{H_{10-11}}$ $\overline{N_{10-11}}$ $\overline{H_{8-9}}$ $\overline{N_{8-9}}$ $\overline{H_{6-7}}$ $\overline{N_{6-7}}$
$\overline{H_{12-13}}$ $\overline{N_{12-13}}$ $\overline{H_{14-15}}$ $\overline{N_{14-15}}$ $\overline{H_{16-17}}$ $\overline{N_{16-17}}$
$\overline{H_{18-19}}$ $\overline{N_{18-19}}$ $\overline{H_{20-21}}$ $\overline{N_{20-21}}$ $\overline{H_{22-23}}$ $\overline{N_{22-23}}$
$\overline{H_{24-25}}$ $\overline{N_{24-25}}$ $\overline{H_{26-27}}$ $\overline{N_{26-27}}$ $\overline{H_{28-29}}$ $\overline{N_{28-29}}$
$\overline{H_{34-35}}$ $\overline{N_{34-35}}$ $\overline{H_{32-33}}$ $\overline{N_{32-33}}$ $\overline{H_{30-31}}$ $\overline{N_{30-31}}$
S_{0-5} S_{6-11} S_{12-17} S_{18-23} S_{24-27} S_{30-35}
Chan. 1 Chan. 2 Chan. 3 Chan. 4 Chan. 5 Chan. 6

Fig. 12.10.1 S Adder Functional Block (Sh 1 of 2)

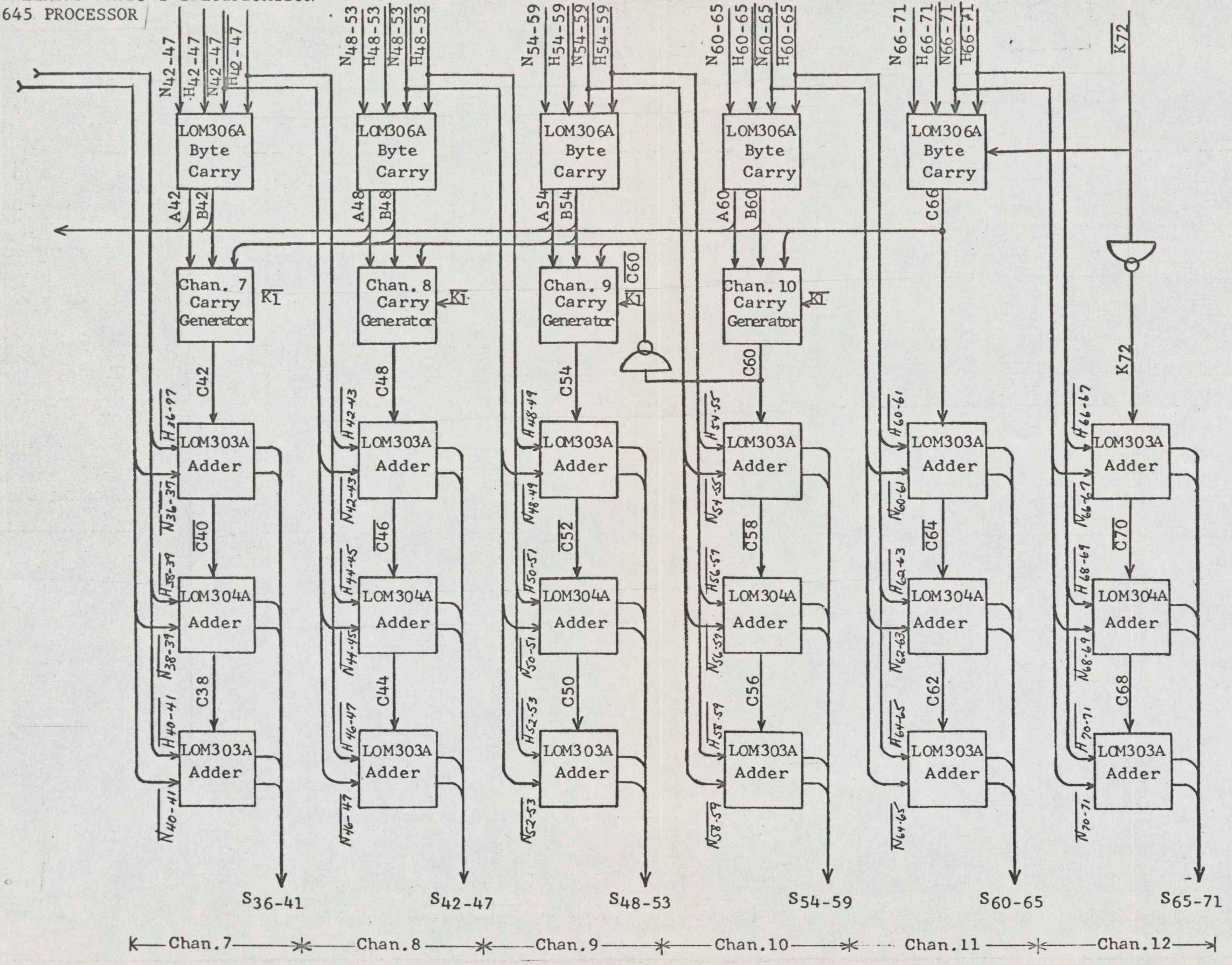

N42-47
H42-47
N42-47
H42-47
N48-53
H48-53
N48-53
H48-53
N54-59
H54-59
N54-59
H54-59
N60-65
H60-65
N60-65
H60-65
N66-71
H66-71
N66-71
H66-71
K72
LOM306A Byte Carry
A42
B42
A48
B48
A54
B54
A60
B60
C66
Chan. 7 Carry Generator
Chan. 8 Carry Generator
Chan. 9 Carry Generator
Chan. 10 Carry Generator
K1
C60
C42
C48
C54
C60
K72
LOM303A Adder
LOM304A Adder
C40
C46
C52
C58
C64
C70
C38
C44
C50
C56
C62
C68
H36-37
N36-37
H38-39
N38-39
H40-41
N40-41
H42-43
N42-43
H44-45
N44-45
H46-47
N46-47
H48-49
N48-49
H50-51
N50-51
H52-53
N52-53
H54-55
N54-55
H56-57
N56-57
H58-59
N58-59
H60-61
N60-61
H62-63
N62-63
H64-65
N64-65
H66-67
N66-67
H68-69
N68-69
H70-71
N70-71
S36-41
S42-47
S48-53
S54-59
S60-65
S65-71
Chan.7
Chan.8
Chan.9
Chan.10
Chan.11
Chan.12

Fig. 12.10.1 S Adder Functional Block

(Sh 2 of 2)

12.10.2 LOOK AHEAD CARRY

This feature of the adder makes the above mentioned speed possible. It consists of the Byte Carry Detector and the Channel Carry Generator.

The function of each Byte Carry Detector module is to determine whether the two byte operands of the next higher numbered channel will produce a carry that should be added to this channel. The outputs, designated A_{6n} and B_{6n}, indicate the result of the byte carry detection as follows:

A_{6n}	B_{6n}	
1	1	an unconditional carry into this channel.
1	0	a carry into this channel only if the next higher numbered channel has a carry into it.
0	X	no carry into this channel.

The Byte Carry Detector for channel n examines the two byte operands entering channel n + 1. If the two byte operands have a sum equal to or greater than 2^6, the A_{6n} and B_{6n} outputs will both be a "1" denoting that a carry is to be entered into channel n. If the two byte operands have a sum exactly equal to $2^6 -1$, the A_{6n} output will be a "1" and the B_{6n} output will be a "0" denoting that a carry is to be entered into channel n only if a carry is entered into channel n + 1. If the two byte operands have a sum less than $2^6 -1$, the A_{6n} output will be a "0" denoting that no carry is to be entered into channel n regardless of the state of B_{6n} or the carry into channel n + 1. There are 12 Byte Carry Detector modules. Channel 12 does not require one as there is no higher numbered channel from which to detect a carry.

12.10.2 - contd

However, a Byte Carry Detector is used to determine if there is a carry out of channel 1 (that is, out of the adder). Figure 12.10.2 shows the logic circuit implementation of the Byte Carry Detector. Pin 41 will be open on all detectors except channel 11 where it will be $\overline{K_{72}}$ and the detector for SC_0 where it will be C_6.

Note that each Byte Carry Detector examines only the operand bytes of channel n + 1 and as such each detector is operating independently of all other detectors. This is the first stage of the S Adder carry look-ahead network. The Channel Carry Generators perform a second and final stage of carry look-ahead.

The function of the Channel Carry Generator is to examine the outputs of its Byte Carry Detector and the outputs of all Byte Carry Detectors of higher numbered channels to determine whether or not a carry is to be inserted into its channel. The outputof the Channel Carry Generator (C_{6n}) will be active if A_{6n} of its Byte Carry Detector is active and either B_{6n} of its detector is active or there is a carry from channel n + 1. This logic is implemented so that the carry C_{6n} can be generated within two logic levels. It should be noted that all carries C_{6n} will be inhibited by K_1 set. There are only 10 Channel Carry Generator modules. Channel 12 derives its carry input from K_{72} and channel 11 derives its carry input from its Byte Carry Detector.

Every sixth carry for the 72-bit addition is produced by the carry look-ahead network. Each carry is then fed into a 6-bit adder which consists of three 2-bit adder modules in cascade.

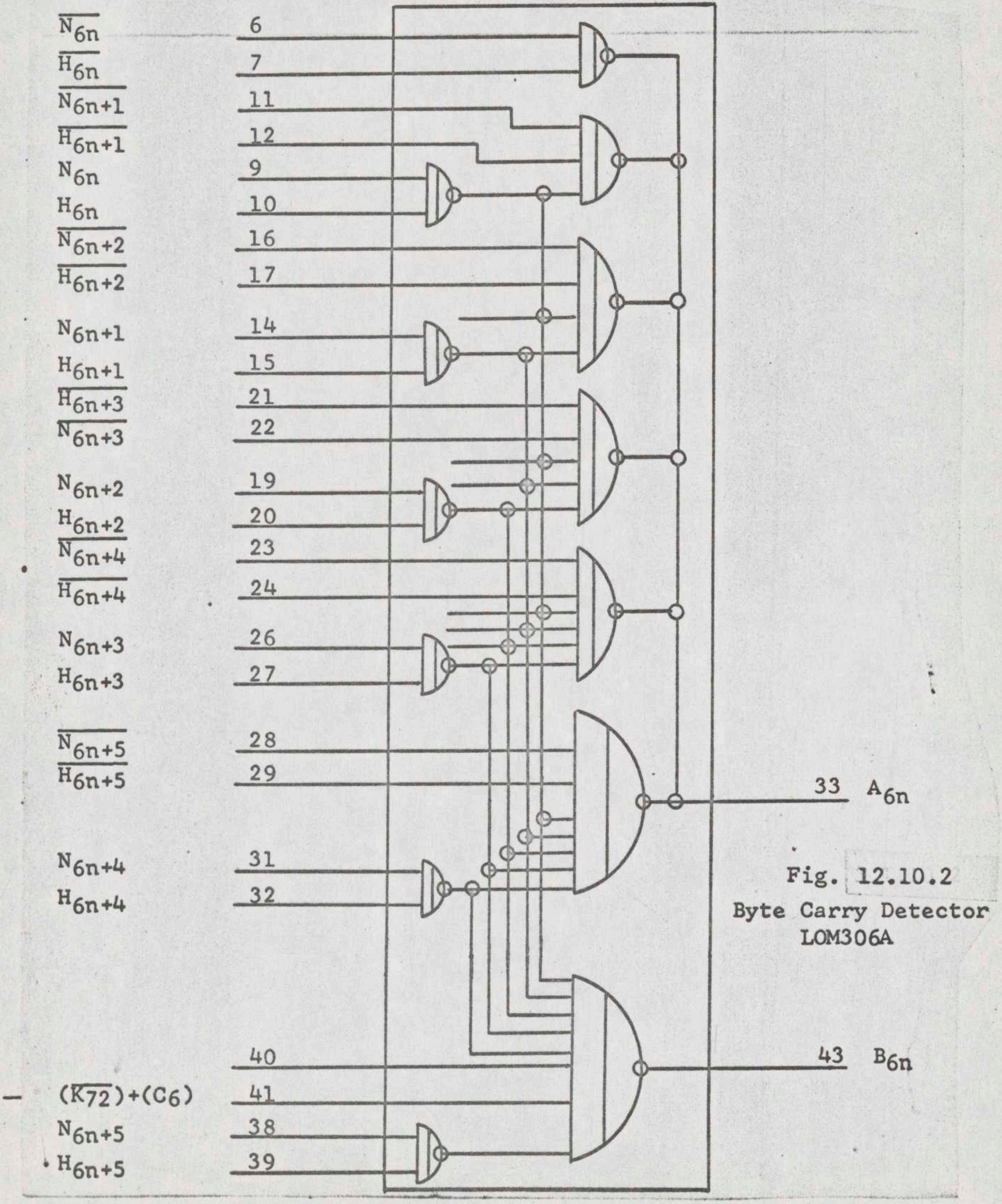

$\overline{N_{6n}}$ 6
$\overline{H_{6n}}$ 7
$\overline{N_{6n+1}}$ 11
$\overline{H_{6n+1}}$ 12
N_{6n} 9
H_{6n} 10
$\overline{N_{6n+2}}$ 16
$\overline{H_{6n+2}}$ 17
N_{6n+1} 14
H_{6n+1} 15
$\overline{H_{6n+3}}$ 21
$\overline{N_{6n+3}}$ 22
N_{6n+2} 19
H_{6n+2} 20
$\overline{N_{6n+4}}$ 23
$\overline{H_{6n+4}}$ 24
N_{6n+3} 26
H_{6n+3} 27
$\overline{N_{6n+5}}$ 28
$\overline{H_{6n+5}}$ 29
33 A_{6n}
N_{6n+4} 31
H_{6n+4} 32
40
43 B_{6n}
$(\overline{K_{72}})+(C6)$ 41
N_{6n+5} 38
H_{6n+5} 39

Fig. 12.10.2
Byte Carry Detector
LOM306A

12.10.3 ADDER MODULES

There are two types of adder modules - LOM303A and LOM304A. Each adder module processes two bits of the two byte operands. The only difference in the two modules is in the carry network. Figure 12.10.3a is a functional block diagram of the LOM303A adder module. This gives an over-all picture of how the module works. Each block will be described in detail.

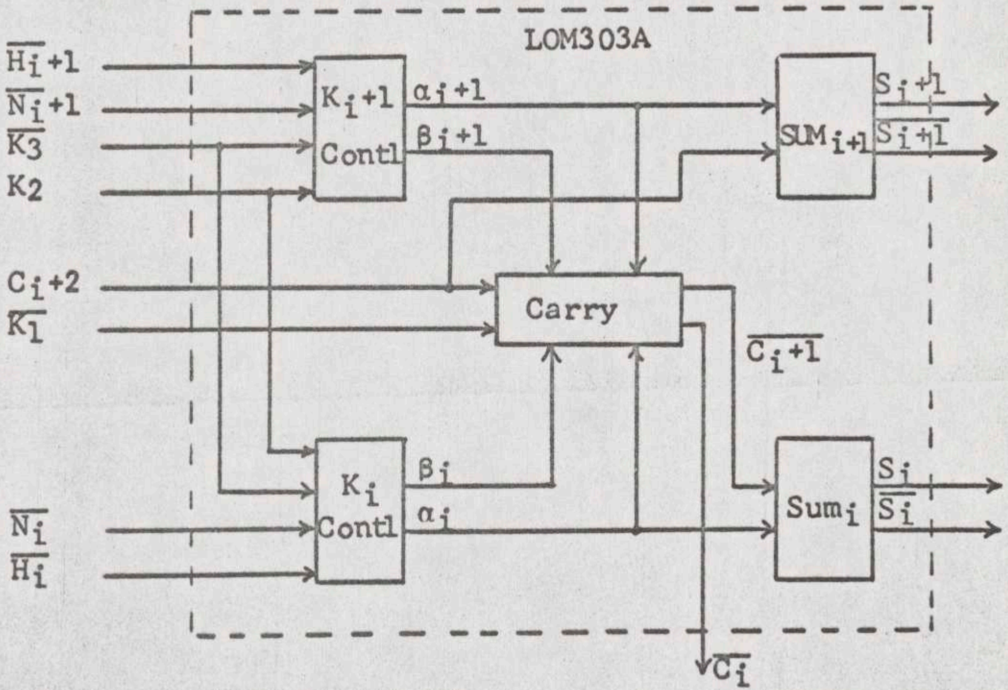

LOM303A
$\overline{H_i}+1$
$\overline{N_i}+1$
$\overline{K_3}$
K_2
K_i+1 Contl
α_i+1
β_i+1
SUM $_{i+1}$
S_i+1
$\overline{S_i+1}$
C_i+2
$\overline{K_1}$
Carry
$\overline{C_i+1}$
K_i Contl
β_i
α_i
$\overline{N_i}$
$\overline{H_i}$
Sum $_i$
S_i
$\overline{S_i}$
$\overline{C_i}$

Fig. 12.10.3a
LM303A Adder, Block Diagram

12.10.3 - contd

It might be wise at this time to discuss the implementation of the K flip-flops. As stated before K_1, K_2, and K_3 determine what function the adder performs. To cause this function to occur in the adder cards it is necessary to take some control signals from the "1" side of the flip-flop and some from the "0" or L side. Below is a table for both the LOM303A and the LOM304A.

K Register			Function
K_1	K_2	K_3	
0	0	0	Add
1	0	0	Exclusive OR
1	1		Inclusive OR
1	0	1	AND

LOM303A	Pin 19	Pin 14	Pin 16
	$\overline{K_1}$	K_2	$\overline{K_3}$
Add	+6	0	+6
Exclusive OR	0	0	+6
Inclusive OR	0	+6	+6
AND	0	0	0

LOM304A	Pin 24	Pin 14	Pin 16
	K_1	K_2	K_3
Add	0	0	+6
Exclusive OR	+6	0	+6
Inclusive OR	+6	+6	+6
AND	+6	0	0

Figure 12.10.3b shows the implementation of the K_1 + 1 control portion of the adder module.

FIG. 12.10.3b LM303A K1+1 Control, Logic Diagram

12.10.3 - contd

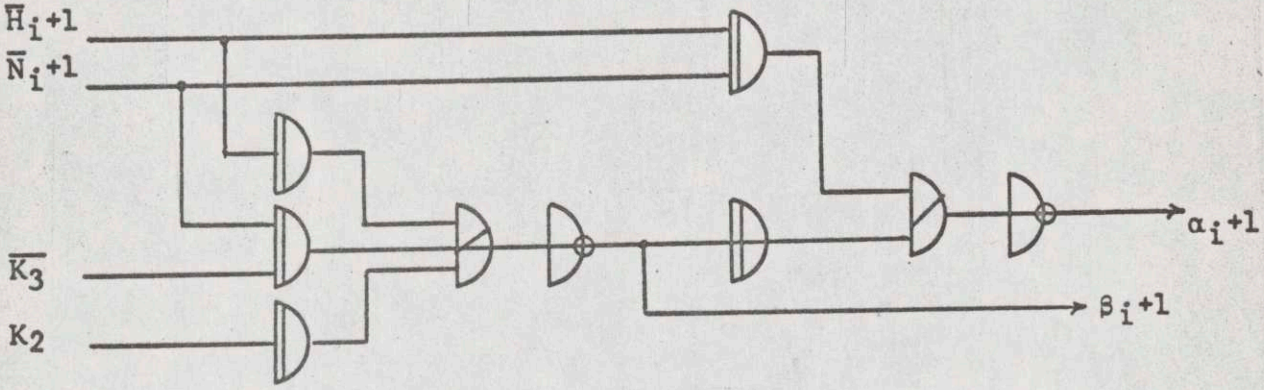


FIG. 12.10.3b LM303A Ki+1 Control, Logic Diagram

The two outputs of this circuit can best be described by use of a truth table.

Operation	H,N	$\overline{H}$,N	H,$\overline{N}$	$\overline{H}$,$\overline{N}$
Add ($\overline{K_1}$, $\overline{K_2}$, $\overline{K_3}$)	$\overline{\alpha_i+1}, \beta_i+1$	$\alpha_i+1, \overline{\beta_i+1}$	$\alpha_i+1, \overline{\beta_i+1}$	$\overline{\alpha_i+1}, \overline{\beta_i+1}$
Exclusive OR (K_1, $\overline{K_2}$, $\overline{K_3}$)	$\overline{\alpha_i+1}, \beta_i+1$	$\alpha_i+1, \overline{\beta_i+1}$	$\alpha_i+1, \overline{\beta_i+1}$	$\overline{\alpha_i+1}, \overline{\beta_i+1}$
Inclusive OR (K_1, K_2, K_3)	$\alpha_i+1, \overline{\beta_i+1}$	$\alpha_i+1, \overline{\beta_i+1}$	$\alpha_i+1, \overline{\beta_i+1}$	$\overline{\alpha_i+1}, \overline{\beta_i+1}$
AND (K_1, $\overline{K_2}$, K_3)	$\overline{\alpha_i+1}, \beta_i+1$	$\alpha_i+1, \overline{\beta_i+1}$	$\overline{\alpha_i+1}, \beta_i+1$	$\overline{\alpha_i+1}, \overline{\beta_i+1}$

The outputs of this circuit become the inputs to the carry and sum circuits. The circuit for α_i and β_i is identical to the one shown above.

Figure 12.10.3c shows the implementation of the $C_i + 1$ carry.

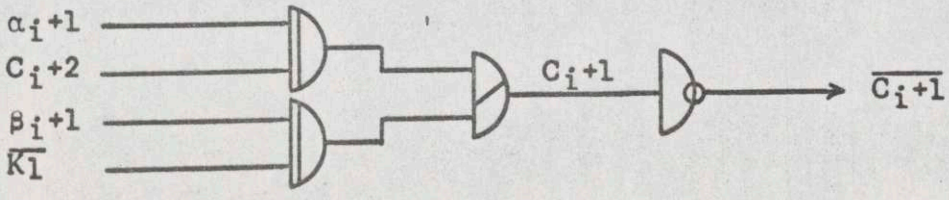


FIG. 12.10.3c LM303A Ci+1 Carry, Logic Diagram

12.10.3 - contd

$C_i + 2$ is the carry out of the next higher number bit position. It can be the carry output of a LOM304A adder module (C_2, C_8, etc.), the carry output of a Channel Carry Detector (C_6, C_{12}, etc.), the carry output of a Byte Carry (C_{66}), or K_{72}. From this diagram it can be seen that if K_1 is set the carry $C_i + 1$ will depend upon receiving a carry in $C_i + 2$, which in turn will depend upon receiving a carry in, etc. all the way back to K_{72}. If K_{72} is reset, as it will be for all non-add operations, no carries will be generated. If an Add operation is in progress $C_i + 1$ will be true if $B_i + 1$ is true or if $\alpha_i + 1$ is true and there is a carry in. Checking the truth tables for the K control reveals that $C_i + 1$ will be true under the following conditions:

a) $(N_i + 1)$ $(H_i + 1)$

b) $(\overline{N_i + 1})$ $(H_i + 1)$ $(C_i + 2)$

c) $(N_i + 1)$ $(\overline{H_i + 1})$ $(C_i + 2)$

$C_i + 1$ is used as an input to the Sum logic for S_i.

Figure 12.10.3d shows the implementation of the C_i carry.

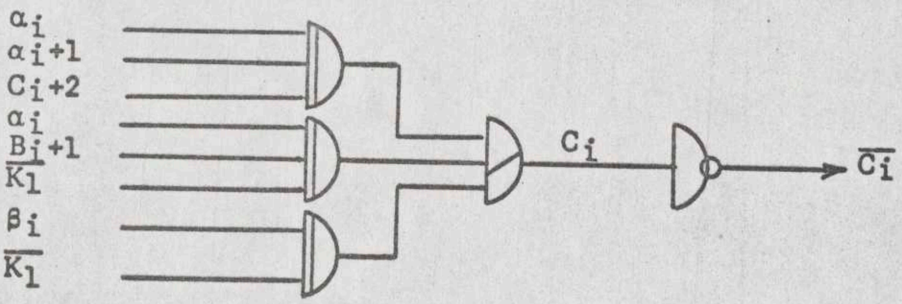


Fig. 12.10.3d

LM303A Ci Carry, Logic Diagram

12.10.3 - contd

For the same reasons as for $C_i + 1$, no carries (C_i) will be generated for non-add operations. For Add operations it was seen that $C_i + 1 = \beta_i + 1 + (\alpha_i + 1)(C_i + 2)$. These same terms are used in the generation of C_i. If they are replaced by $C_i + 1$ then $C_i = (\alpha_i)(C_i + 1) + \beta_i$. In effect the $C_i + 1$ network is reproduced and α_i added. This was done to speed up the generation of C_i. For an Add operation C_i will be true under the following conditions:

a) $(N_i)(H_i)$

b) $(\overline{N_i})(H_i)(C_i + 1)$

c) $(N_i)(\overline{H_i})(C_i + 1)$

$\overline{C_i}$ is the carry output of the LOM303A adder module. It should be noted that the LOM303A requires a true carry input and generates the false carry output.

Figure 12.10.3e shows the implementation of the $\text{Sum}_i + 1$ logic.

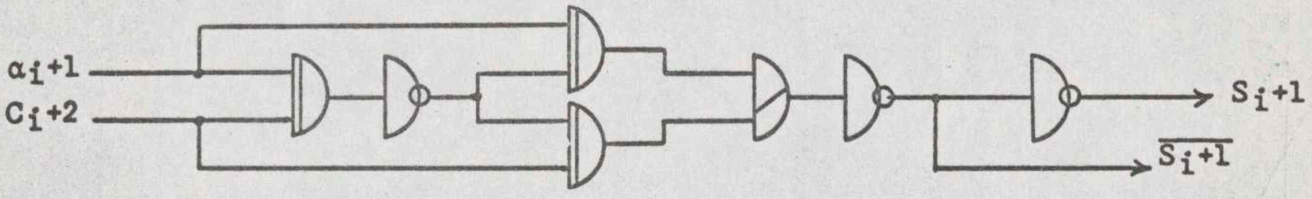


FIG. 12.10.3e LM303A Sum_i+1, Logic Diagram

From this it can be seen that $S_i + 1$ will be true only if the two inputs ($\alpha_i + 1$ and $C_i + 2$) are not the same. By using the previous information a truth table can be constructed for all operations.

12.10.3 - contd

	H_i+1, N_i+1	$\overline{H_i+1}, N_i+1$	$H_i+1\overline{N_i+1}$	$\overline{H_i+1}, \overline{N_i+1}$
Add $\overline{C_i+2}$	$\overline{S_i+1}$	S_i+1	S_i+1	$\overline{S_i+1}$
Add C_i+2	S_i+1	$\overline{S_i+1}$	$\overline{S_i+1}$	S_i+1
Exclusive OR $\overline{C_i+2}$	$\overline{S_i+1}$	S_i+1	S_i+1	$\overline{S_i+1}$
Inclusive OR $\overline{C_i+2}$	S_i+1	S_i+1	S_i+1	$\overline{S_i+1}$
AND $\overline{C_i+2}$	$\overline{S_i+1}$	S_i+1	$\overline{S_i+1}$	$\overline{S_i+1}$

At first glance it would seem that the output will be wrong for the AND operation, however, during an AND operation the ones complement of the operand from memory is stored in H to make the output of the Adder correct.

The Sum_i logic is the same as in Fig. 12.10.3e except that the inputs will be a_i and $\overline{C_i+1}$ and the outputs will be reversed. A truth table can be constructed for this logic. It will be the same as the one for S_i+1.

The LOM304A adder module is similar to the LOM303A adder module. It requires a false carry input and it produces a true carry output. The circuitry differs only in the carry logic. Figure 12.10.3f is a functional block diagram of the LOM304A module.

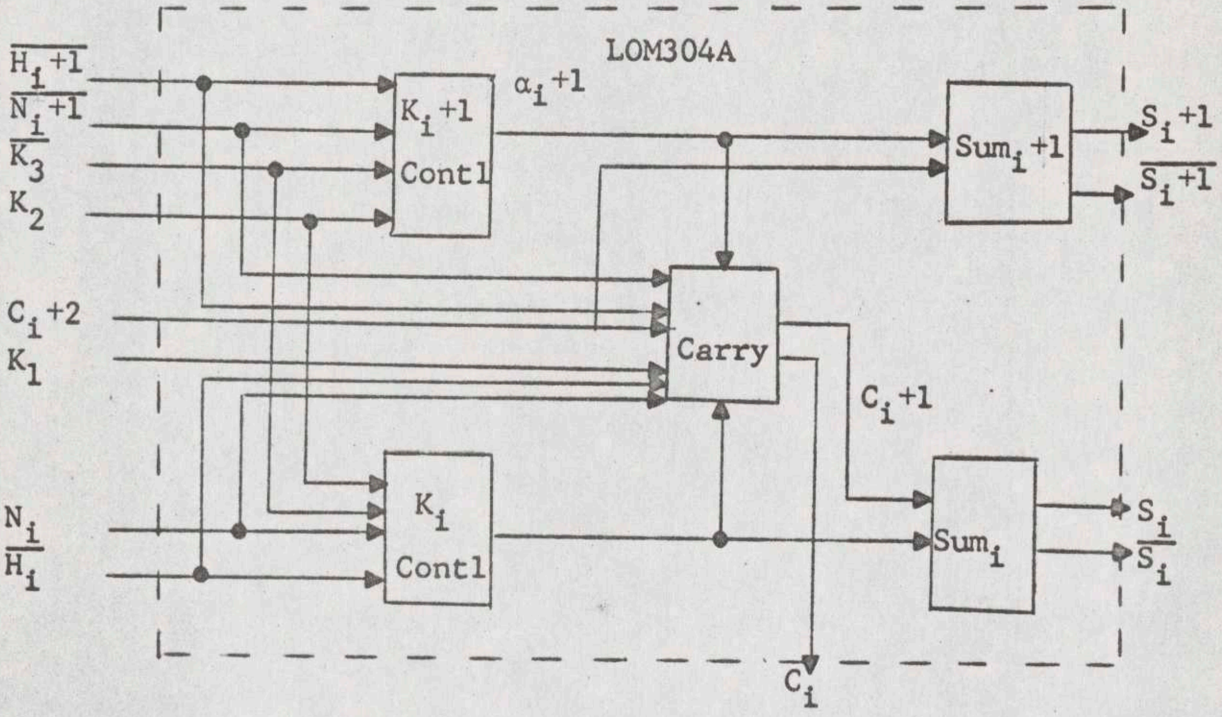

LOM304A
$\overline{H_i+1}$
$\overline{N_i+1}$
$\overline{K_3}$
K_2
α_i+1
K_i+1 Contl
Sum_i+1
S_i+1
$\overline{S_i+1}$
C_i+2
K_1
Carry
C_i+1
N_i
$\overline{H_i}$
K_i Contl
Sum_i
S_i
$\overline{S_i}$
C_i

FIG. 12.10.3f LM304A, Block Diagram

The K control logic is identical to the K control logic of the LOM303A module. The β output is not used in the LOM304A. The truth table for the LOM303A will also apply for this module.

Figure 12.10.3g shows the implementation of the C_i+1 carry.

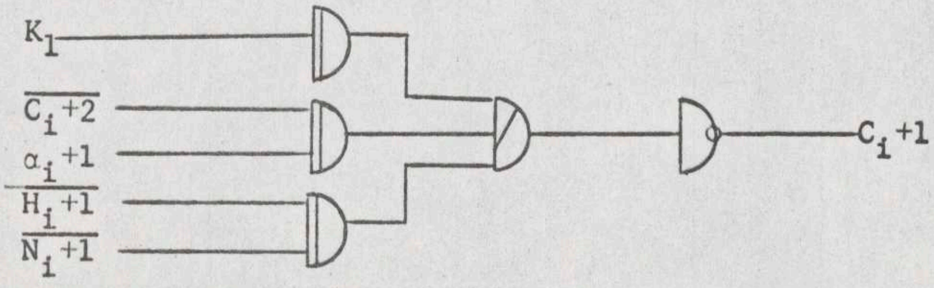

K_1
$\overline{C_i+2}$
α_i+1
$\overline{H_i+1}$
$\overline{N_i+1}$
C_i+1

FIG. 12.10.3g LM304A, Ci+1 Carry, Logic Diagram

12.10.3 - contd

$\overline{C_i + 2}$ is the carry out of the next higher numbered bit position. It will always be the output of a LOM303A module. From this diagram it can be seen that if K_1 is set $C_i + 1$ will always be false. If K_1 is reset, an Add operation, $C_i + 1$ will be true under the following conditions:

a) $(H_i + 1)\ (N_i + 1)$

b) $(\overline{H_i + 1})\ (N_i + 1)\ (C_i + 2)$

c) $(H_i + 1)\ (\overline{N_i + 1})\ (C_i + 2)$

$C_i + 1$ is used as input to the Sum logic for $S_i + 1$.

Figure 12.10.3h shows the implementation of the C_i carry.

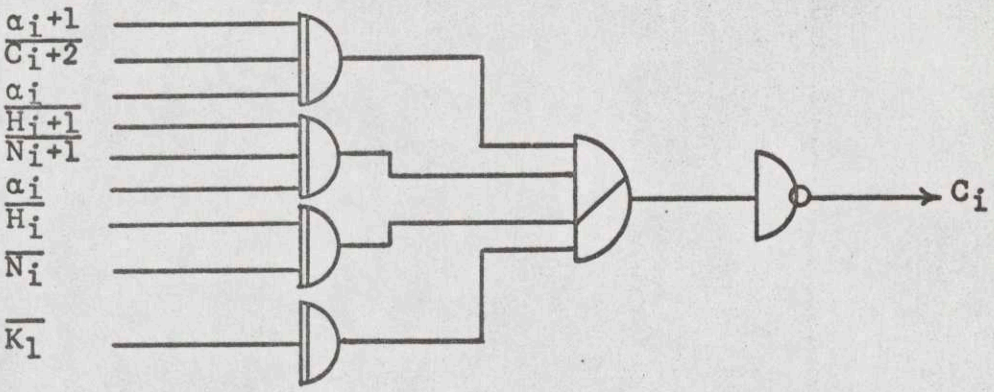

$\alpha_i + 1$
$\overline{C_i + 2}$
α_i
$\overline{H_i + 1}$
$\overline{N_i + 1}$
α_i
$\overline{H_i}$
$\overline{N_i}$
$\overline{K_1}$
C_i

FIG. 12.10.3h LM304A Ci Carry, Logic Diagram

If K_1 is set (all non-add operations) C_i will always be false. During Add operations if any one of the other three gates is satisfied C_i will be false. The truth table follows.

12.10.3 - contd

		(α_i+1)	(α_i+1)	
If C_i+2	H_i+1, N_i+1	H_i+1, N_i+1	H_i+1, N_i+1	H_i+1, N_i+1
$H_i\ N_i$	C_i	C_i	C_i	C_i
$\overline{H_i}\ N_i(\alpha_i)$	C_i	C_i	C_i	$\overline{C_i}$
$H_i\ \overline{N_i}(\alpha_i)$	C_i	C_i	C_i	$\overline{C_i}$
$\overline{H_i}\ \overline{N_i}$	$\overline{C_i}$	$\overline{C_i}$	$\overline{C_i}$	$\overline{C_i}$
If $\overline{C_i+2}$				
$H_i\ N_i$	C_i	C_i	C_i	C_i
$\overline{H_i}\ N_i\ (\alpha_i)$	C_i	$\overline{C_i}$	$\overline{C_i}$	$\overline{C_i}$
$H_i\ \overline{N_i}\ (\alpha_i)$	C_i	$\overline{C_i}$	$\overline{C_i}$	$\overline{C_i}$
$\overline{H_i}\ \overline{N_i}$	$\overline{C_i}$	$\overline{C_i}$	$\overline{C_i}$	$\overline{C_i}$

C_i is the carry output of the LOM304A adder module. This module requires a false carry input and generates a true carry output.

The S_i and S_i +1 logic is identical to the LOM303A logic except that the output points are reversed. The same truth table applies here.

12.11 FLOW DIAGRAMS AND ALGORITHM DESCRIPTION

Figure 12.11	Flow Diagram for MPY (Integer Multiply)
Figure 12.11a	Flow Diagram for DIV (Interger Divide)
Figure 12.11b	Flow Diagram for BCD (Binary to Binary Coded Decimal)
Figure 12.11c	Flow Diagram for FAD (Floating Add)

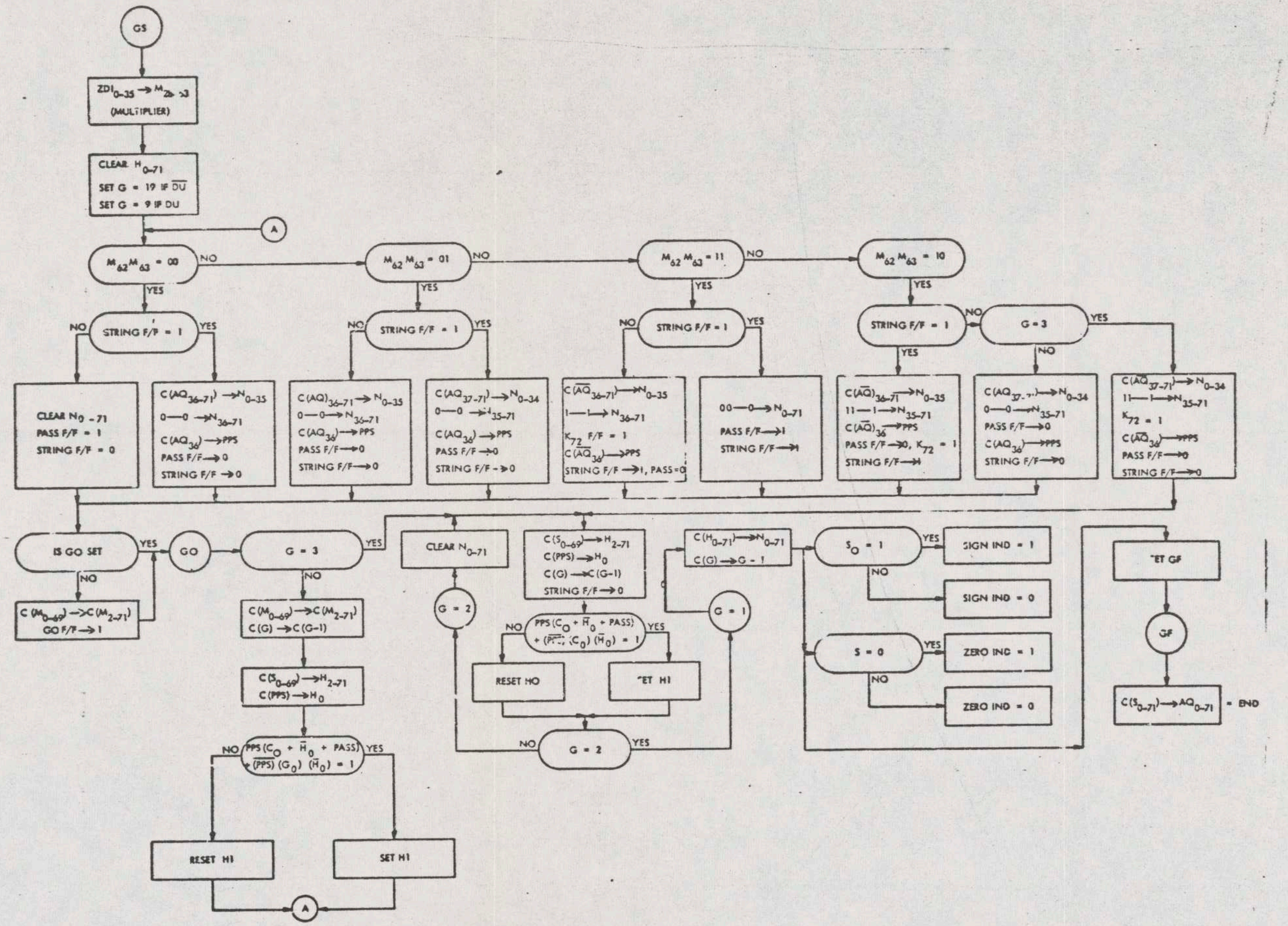

GS
$ZDI_{0-35} \rightarrow M_{28-63}$ (MULTIPLIER)
CLEAR H_{0-71}
SET G = 19 IF $\overline{DU}$
SET G = 9 IF DU
A
$M_{62}M_{63}$ = 00
NO
YES
$M_{62}M_{63}$ = 01
$M_{62}M_{63}$ = 11
$M_{62}M_{63}$ = 10
STRING F/F = 1
G = 3
CLEAR N_{0-71}
PASS F/F = 1
STRING F/F = 0
$C(AQ_{36-71}) \rightarrow N_{0-35}$
$0 \rightarrow 0 \rightarrow N_{36-71}$
$C(AQ_{36}) \rightarrow PPS$
PASS F/F → 0
STRING F/F → 0
$C(AQ)_{36-71} \rightarrow N_{0-35}$
$0 \rightarrow 0 \rightarrow N_{36-71}$
$C(AQ_{36}) \rightarrow PPS$
PASS F/F → 0
STRING F/F → 0
$C(AQ_{37-71}) \rightarrow N_{0-34}$
$0 \rightarrow 0 \rightarrow N_{35-71}$
$C(AQ_{36}) \rightarrow PPS$
PASS F/F → 0
STRING F/F → 0
$C(\overline{AQ}_{36-71}) \rightarrow N_{0-35}$
$1 \rightarrow 1 \rightarrow N_{36-71}$
K_{72} F/F = 1
$C(\overline{AQ}_{36}) \rightarrow PPS$
STRING F/F → 1, PASS = 0
$00 \rightarrow 0 \rightarrow N_{0-71}$
PASS F/F → 1
STRING F/F → 1
$C(\overline{AQ})_{36-71} \rightarrow N_{0-35}$
$11 \rightarrow 1 \rightarrow N_{35-71}$
$C(\overline{AQ})_{36} \rightarrow PPS$
PASS F/F → 0, K_{72} = 1
STRING F/F → 1
$C(AQ_{37-71}) \rightarrow N_{0-34}$
$0 \rightarrow 0 \rightarrow N_{35-71}$
PASS F/F → 0
$C(AQ_{36}) \rightarrow PPS$
STRING F/F → 0
$C(\overline{AQ}_{37-71}) \rightarrow N_{0-34}$
$11 \rightarrow 1 \rightarrow N_{35-71}$
K_{72} = 1
$C(\overline{AQ}_{36}) \rightarrow PPS$
PASS F/F → 0
STRING F/F → 0
IS GO SET
GO
$C(M_{0-69}) \rightarrow C(M_{2-71})$
GO F/F → 1
$C(M_{0-69}) \rightarrow C(M_{2-71})$
$C(G) \rightarrow C(G-1)$
$C(S_{0-69}) \rightarrow H_{2-71}$
$C(PPS) \rightarrow H_0$
$PPS(C_0 + \overline{H}_0 + PASS) + (\overline{PPS})(G_0)(\overline{H}_0) = 1$
RESET H1
SET H1
CLEAR N_{0-71}
G = 2
$C(S_{0-69}) \rightarrow H_{2-71}$
$C(PPS) \rightarrow H_0$
$C(G) \rightarrow C(G-1)$
STRING F/F → 0
$PPS(C_0 + \overline{H}_0 + PASS) + (\overline{PPS})(C_0)(\overline{H}_0) = 1$
RESET HO
SET H1
G = 2
G = 1
$C(H_{0-71}) \rightarrow N_{0-71}$
$C(G) \rightarrow G - 1$
S_0 = 1
SIGN IND = 1
SIGN IND = 0
S = 0
ZERO IND = 1
ZERO IND = 0
SET GF
GF
$C(S_{0-71}) \rightarrow AQ_{0-71}$ = END

FIG. 12.11 FLOW DIAGRAM FOR MPY (INTEGER MULTIPLY)

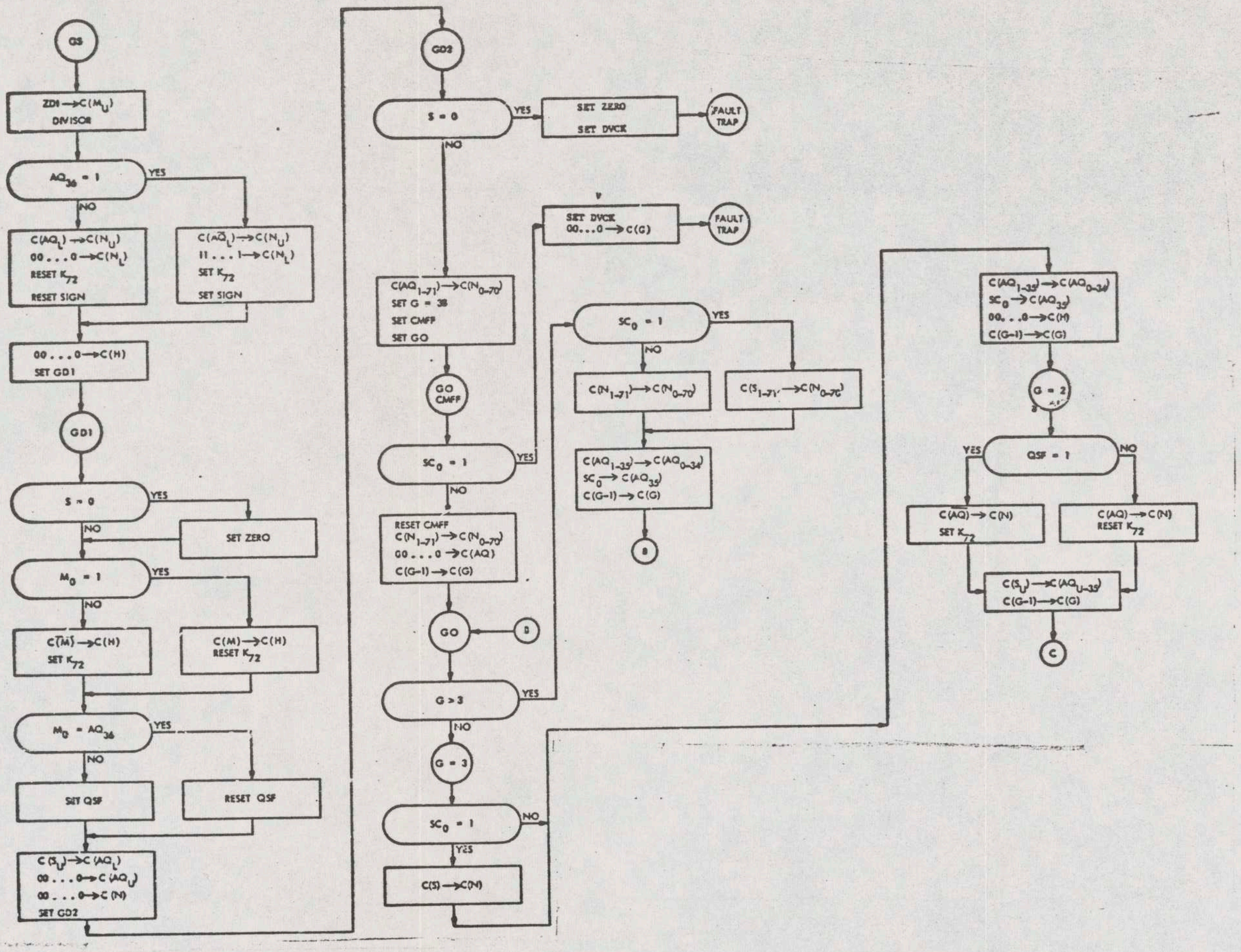

GS
ZDI → $C(M_U)$
DIVISOR
AQ_{36} = 1
YES
NO
$C(AQ_L)$ → $C(N_U)$
00 . . . 0 → $C(N_L)$
RESET K_{72}
RESET SIGN
$C(\overline{AQ_L})$ → $C(N_U)$
11 . . . 1 → $C(N_L)$
SET K_{72}
SET SIGN
00 . . . 0 → C(H)
SET GD1
GD1
S = 0
YES
NO
SET ZERO
M_0 = 1
YES
NO
$C(\overline{M})$ → C(H)
SET K_{72}
C(M) → C(H)
RESET K_{72}
M_0 = AQ_{36}
YES
NO
SET QSF
RESET QSF
$C(S_U)$ → $C(AQ_L)$
00 . . . 0 → $C(AQ_U)$
00 . . . 0 → C(N)
SET GD2
GD2
S = 0
YES
SET ZERO
SET DVCK
FAULT TRAP
NO
SET DVCK
00...0 → C(G)
FAULT TRAP
$C(AQ_{1-71})$ → $C(N_{0-70})$
SET G = 38
SET CMFF
SET GO
GO CMFF
SC_0 = 1
YES
NO
RESET CMFF
$C(N_{1-71})$ → $C(N_{0-70})$
00 . . . 0 → C(AQ)
C(G-1) → C(G)
GO
B
G > 3
YES
NO
G = 3
SC_0 = 1
NO
YES
C(S) → C(N)
SC_0 = 1
YES
NO
$C(N_{1-71})$ → $C(N_{0-70})$
$C(S_{1-71})$ → $C(N_{0-70})$
$C(AQ_{1-35})$ → $C(AQ_{0-34})$
SC_0 → $C(AQ_{35})$
C(G-1) → C(G)
B
$C(AQ_{1-35})$ → $C(AQ_{0-34})$
SC_0 → $C(AQ_{35})$
00. . .0 → C(H)
C(G-1) → C(G)
G = 2
QSF = 1
YES
NO
$C(\overline{AQ})$ → C(N)
SET K_{72}
C(AQ) → C(N)
RESET K_{72}
$C(S_U)$ → $C(AQ_{U-35})$
C(G-1) → C(G)
C

FIG. 12.11a FLOW DIAGRAM FOR DIV (INTEGER DIVIDE)

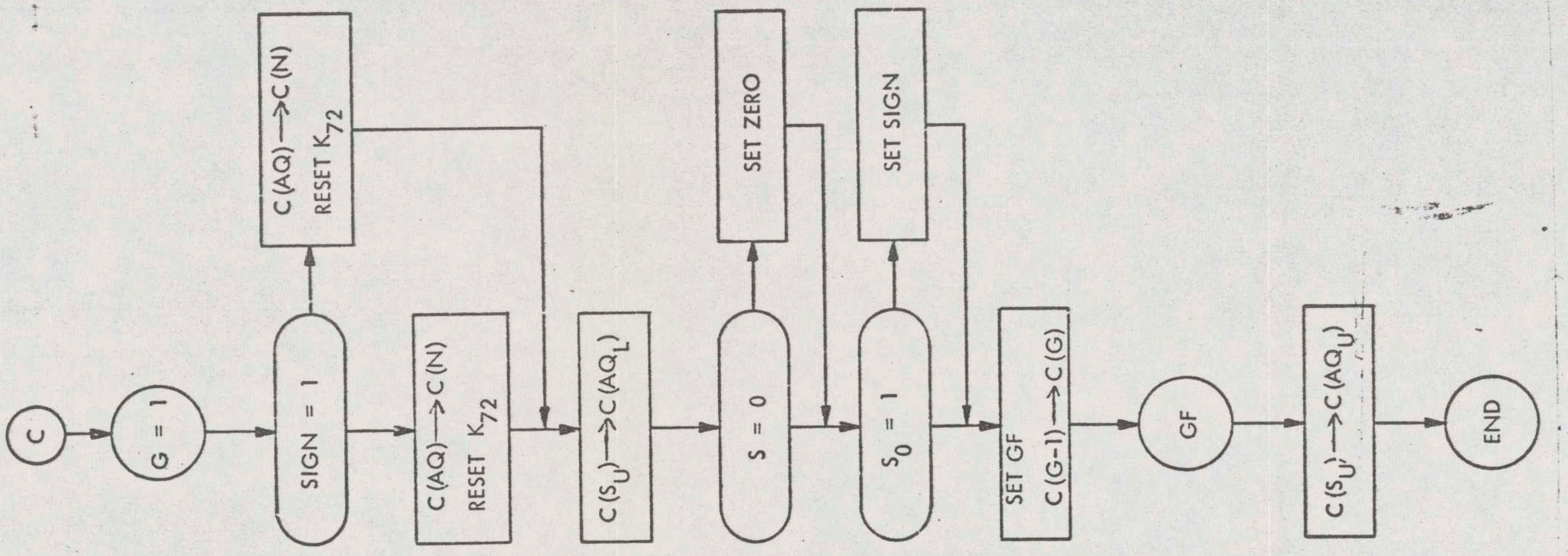

C
G = 1
SIGN = 1
C(AQ) ⟶ C(N) RESET K_{72}
C(AQ) ⟶ C(N) RESET K_{72}
C(S_U) ⟶ C(AQ_L)
S = 0
SET ZERO
S_0 = 1
SET SIGN
SET GF C(G-1) ⟶ C(G)
GF
C($S_{U'}$) ⟶ C(AQ_U)
END

FIG. 12.11a FLOW DIAGRAM FOR DIV (INTEGER DIVIDE)

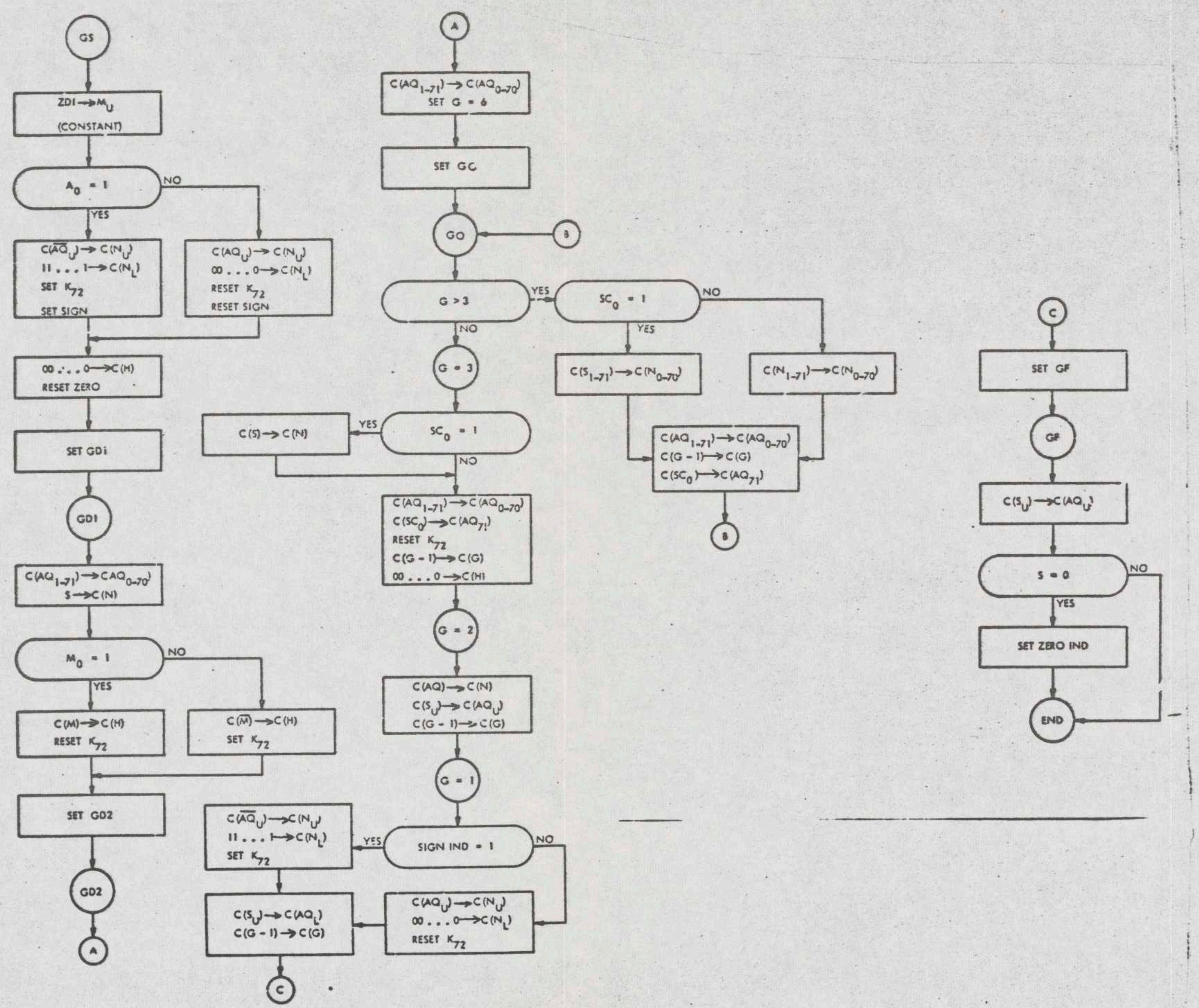

GS
ZDI → M_U
(CONSTANT)
A_0 = 1
NO
YES
$C(\overline{AQ}_U) \rightarrow C(N_U)$
11 . . . 1 → $C(N_L)$
SET K_{72}
SET SIGN
$C(AQ_U) \rightarrow C(N_U)$
00 . . . 0 → $C(N_L)$
RESET K_{72}
RESET SIGN
00 . . . 0 → C(H)
RESET ZERO
SET GD1
GD1
$C(AQ_{1-71}) \rightarrow C(AQ_{0-70})$
S → C(N)
M_0 = 1
NO
YES
C(M) → C(H)
RESET K_{72}
$C(\overline{M}) \rightarrow C(H)$
SET K_{72}
SET GD2
GD2
A
A
$C(AQ_{1-71}) \rightarrow C(AQ_{0-70})$
SET G = 6
SET GC
GO
B
G > 3
YES
NO
G = 3
SC_0 = 1
YES
NO
C(S) → C(N)
$C(AQ_{1-71}) \rightarrow C(AQ_{0-70})$
$C(SC_0) \rightarrow C(AQ_{71})$
RESET K_{72}
C(G - 1) → C(G)
00 . . . 0 → C(H)
G = 2
C(AQ) → C(N)
$C(S_U) \rightarrow C(AQ_U)$
C(G - 1) → C(G)
G = 1
SIGN IND = 1
YES
NO
$C(\overline{AQ}_U) \rightarrow C(N_U)$
11 . . . 1 → $C(N_L)$
SET K_{72}
$C(AQ_U) \rightarrow C(N_U)$
00 . . . 0 → $C(N_L)$
RESET K_{72}
$C(S_U) \rightarrow C(AQ_L)$
C(G - 1) → C(G)
C
SC_0 = 1
NO
YES
$C(S_{1-71}) \rightarrow C(N_{0-70})$
$C(N_{1-71}) \rightarrow C(N_{0-70})$
$C(AQ_{1-71}) \rightarrow C(AQ_{0-70})$
C(G - 1) → C(G)
$C(SC_0) \rightarrow C(AQ_{71})$
B
C
SET GF
GF
$C(S_U) \rightarrow C(AQ_U)$
S = 0
NO
YES
SET ZERO IND
END

FIG. 12.11b FLOW DIAGRAM FOR BCD (BINARY TO BINARY CODED DECIMAL)

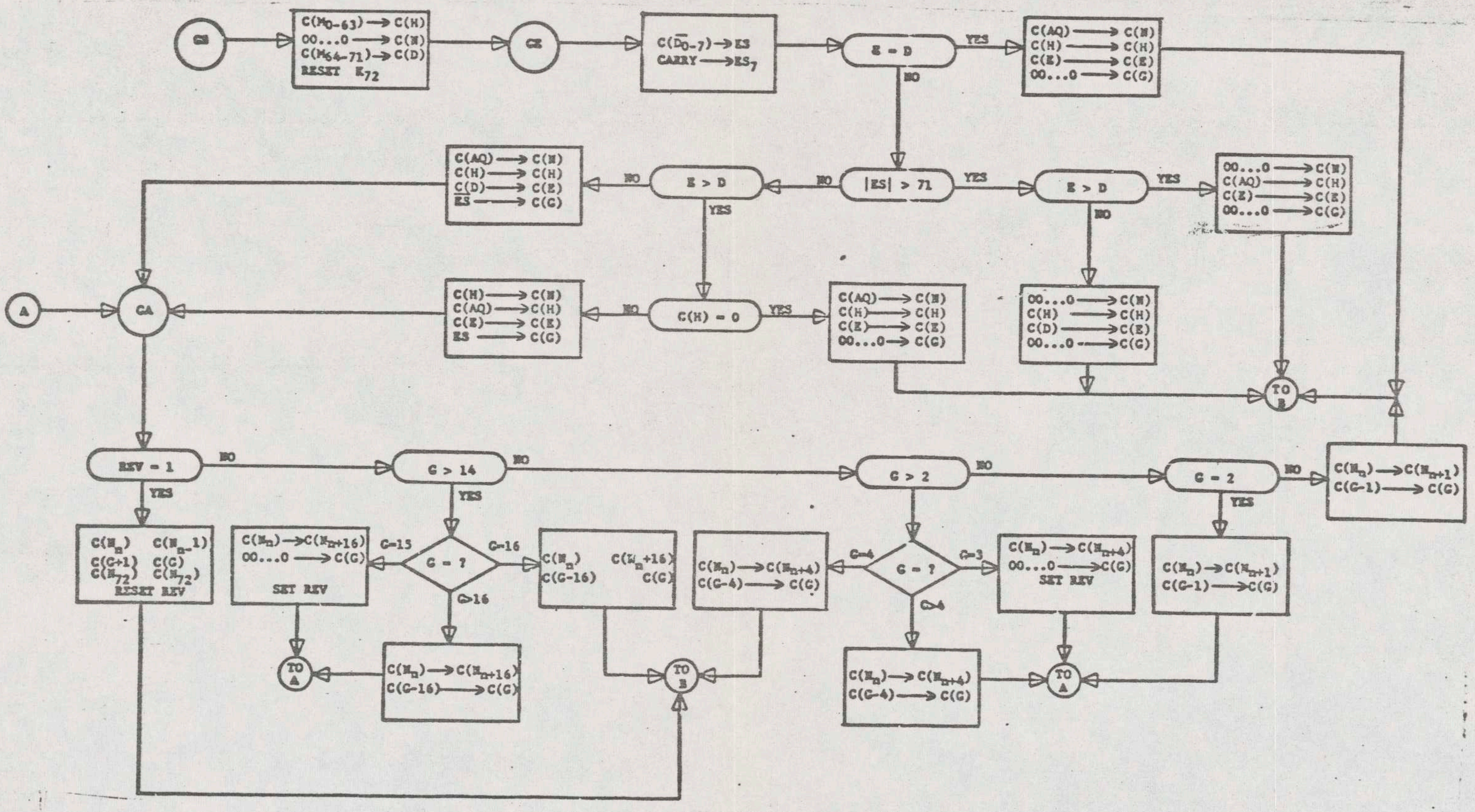

GS
C(M0-63) → C(H)
00...0 → C(N)
C(M64-71) → C(D)
RESET E72
GE
C($\overline{D}$0-7) → ES
CARRY → ES7
E = D
YES
NO
C(AQ) → C(N)
C(H) → C(H)
C(E) → C(E)
00...0 → C(G)
|ES| > 71
YES
NO
E > D
YES
00...0 → C(N)
C(AQ) → C(H)
C(E) → C(E)
00...0 → C(G)
NO
00...0 → C(N)
C(H) → C(H)
C(D) → C(E)
00...0 → C(G)
E > D
NO
YES
C(AQ) → C(N)
C(H) → C(H)
C(D) → C(E)
ES → C(G)
C(H) = 0
NO
YES
C(H) → C(N)
C(AQ) → C(H)
C(E) → C(E)
ES → C(G)
C(AQ) → C(N)
C(H) → C(H)
C(E) → C(E)
00...0 → C(G)
TO B
A
CA
REV = 1
NO
YES
C(Nn) C(Nn-1)
C(G+1) C(G)
C(N72) C(N72)
RESET REV
G > 14
NO
YES
G = ?
G=15
G=16
G>16
C(Nn) → C(Nn+16)
00...0 → C(G)
SET REV
C(Nn) C(Nn+16)
C(G-16) C(G)
C(Nn) → C(Nn+16)
C(G-16) → C(G)
TO A
TO B
G > 2
NO
YES
G = ?
G=4
G=3
G>4
C(Nn) → C(Nn+4)
C(G-4) → C(G)
C(Nn) → C(Nn+4)
00...0 → C(G)
SET REV
C(Nn) → C(Nn+4)
C(G-4) → C(G)
TO A
G = 2
NO
YES
C(Nn) → C(Nn+1)
C(G-1) → C(G)
C(Nn) → C(Nn+1)
C(G-1) → C(G)

FIG. 12.11c FLOW DIAGRAM FOR FAD (FLOATING ADD)

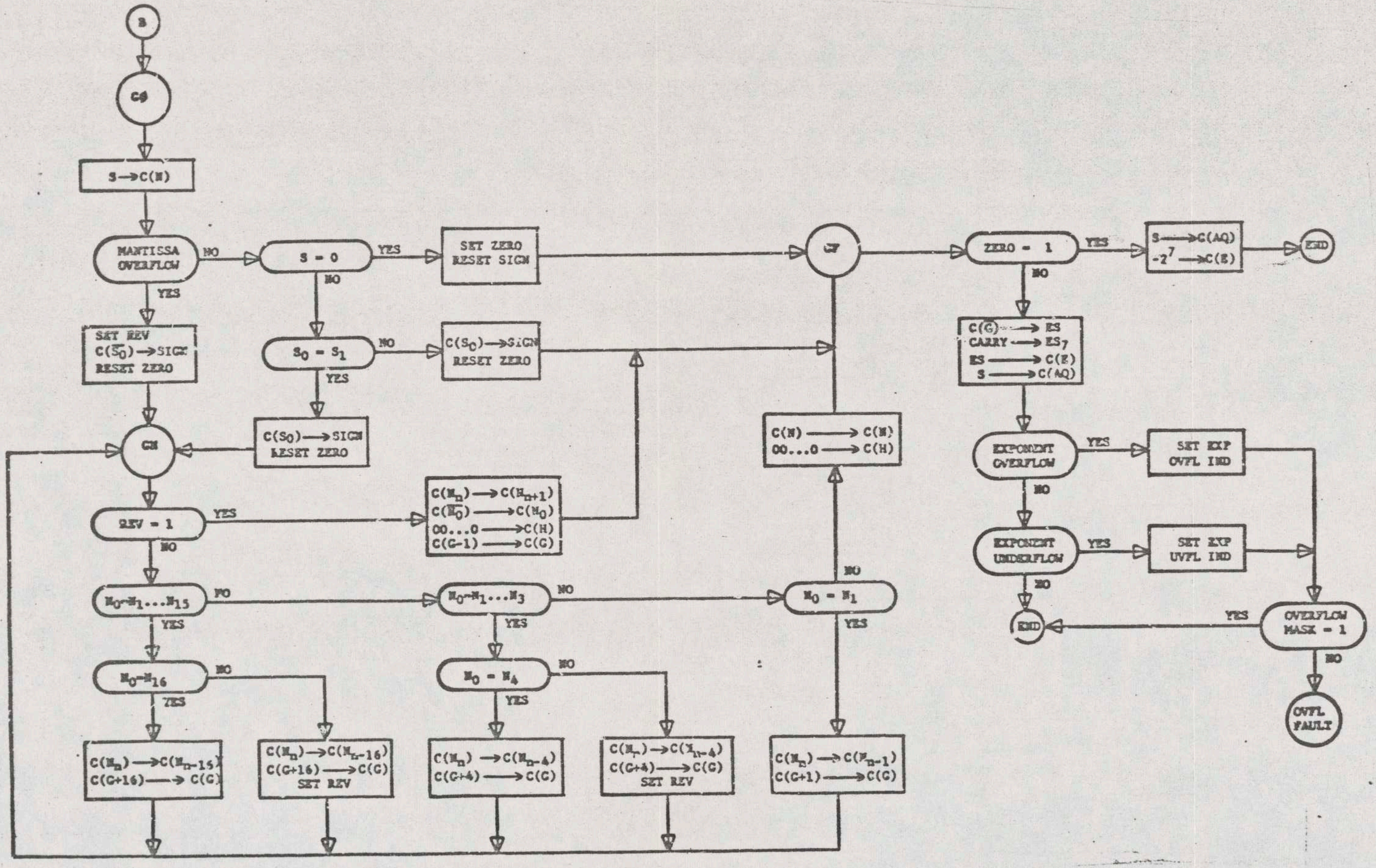

B
GΦ
$S \to C(M)$
MANTISSA OVERFLOW
NO
YES
$S = 0$
YES
NO
SET ZERO RESET SIGN
SET REV
$C(\overline{S_0}) \to$ SIGN
RESET ZERO
$S_0 = S_1$
NO
YES
$C(S_0) \to$ SIGN
RESET ZERO
$C(S_0) \to$ SIGN
RESET ZERO
GN
REV = 1
YES
NO
$C(M_n) \to C(M_{n+1})$
$C(\overline{M_0}) \to C(M_0)$
$00\ldots0 \to C(H)$
$C(G-1) \to C(G)$
$M_0 = M_1 \ldots M_{15}$
NO
YES
$M_0 = M_1 \ldots M_3$
NO
YES
$M_0 = M_1$
NO
YES
$M_0 = M_{16}$
NO
YES
$M_0 = M_4$
NO
YES
$C(M_n) \to C(M_{n-15})$
$C(G+16) \to C(G)$
$C(M_n) \to C(M_{n-16})$
$C(G+16) \to C(G)$
SET REV
$C(M_n) \to C(M_{n-4})$
$C(G+4) \to C(G)$
$C(M_n) \to C(M_{n-4})$
$C(G+4) \to C(G)$
SET REV
$C(M_n) \to C(M_{n-1})$
$C(G+1) \to C(G)$
$C(M) \to C(M)$
$00\ldots0 \to C(H)$
GF
ZERO = 1
YES
NO
$S \to C(AQ)$
$-2^7 \to C(E)$
END
$C(G) \to ES$
$CARRY \to ES_7$
$ES \to C(E)$
$S \to C(AQ)$
EXPONENT OVERFLOW
YES
NO
SET EXP OVFL IND
EXPONENT UNDERFLOW
YES
NO
SET EXP UVFL IND
END
YES
OVERFLOW MASK = 1
NO
OVFL FAULT

FIG. 12.11c FLOW DIAGRAM FOR FAD (FLOATING ADD)

13.0 MAINTENANCE

The whole philosophy of maintenance of the GE-645 Prototype Processor is still to be determined. This information will evolve out of problems encountered during the debug and checkout phases of the hardware and T&D programs and evolution at the prototype site. Until that time, the information available on the Processor Maintenance Panel in Section 3.0 is the only information pertaining to maintenance in this EPS.

COMPANY CONFIDENTIAL

GENERAL ELECTRIC COMPANY
SPECIAL INFORMATION PRODUCTS DEPARTMENT
PHOENIX, ARIZONA

Spec. No. M50EB00107

Cont. on Sh. Final Sh. No. 13-2

TITLE: ENGINEERING PRODUCT SPECIFICATION
GE-645 PROTOTYPE PROCESSOR

RELEASES AND APPROVALS for: Sections 7 through 13.

This document was prepared by the GE-645 Project P. & C.E., General Electric Computer Equipment Department, Phoenix, Arizona. In the process of preparing this document, the GE-645 Project has been transferred and is now the responsibility of GE-645 Equipment Design, ACP Section, Special Information Products Department, Phoenix, Arizona.

Prepared by: Walter B Riordan — March 29 1967
Technical Writer - 645 System Integration Unit — DATE

Reviewed by: Jaime Calle — March 29, 1967
~~Project Engineer~~ - 645 Processor and Monitor Device Unit — DATE

Approved by: Robert H. Morrison — April 4, 1967
Manager - 645 Processor and Monitor Device Unit — DATE

April 10, 1967
Manager - 645 Equipment Design — DATE